Scalable and Trustworthy Learning in Heterogeneous Networks

Tian Li

CMU-CS-23-127

August 2023

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Virginia Smith (Chair)
Tianqi Chen
Ameet Talwalkar
H. Brendan McMahan (Google Research)
Dawn Song (UC Berkeley)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored in part by the National Science Foundation grant IIS1838017, a Carnegie Bosch Institute Research Award, a Google Faculty Award, a Facebook Faculty Award, the Private AI Collaborative Research Institute, and the CONIX Research Center.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: distributed optimization, trustworthy learning, federated learning

To my parents.

Abstract

Developing machine learning models heavily relies on access to data. To build a responsible data economy and protect data ownership, it is crucial to enable learning models from separate, heterogeneous data sources without centralization. Federated learning (FL) aims to train models collectively across massive remote devices or isolated organizations, while keeping user data local. However, federated networks introduce a number of challenges beyond traditional distributed learning scenarios. While FL has shown great promise for enabling edge applications, current FL systems are hindered by several constraints. In addition to being accurate, federated methods must scale to potentially massive and heterogeneous networks of devices, and must exhibit trustworthy behavior—addressing pragmatic concerns related to issues such as fairness, robustness, and user privacy.

In this thesis, we aim to address the practical challenges of federated learning in a principled fashion. We study how heterogeneity lies at the center of the constraints of federated learning—not only affecting the accuracy of the models, but also competing with other critical metrics such as fairness, robustness, and privacy. To address these metrics, we develop new, scalable learning objectives and algorithms that rigorously account for and address sources of heterogeneity. In particular, in terms of accuracy, we propose novel federated optimization frameworks with convergence guarantees under realistic heterogeneity assumptions. In terms of trustworthiness, we develop and analyze fair learning objectives which offer flexible fairness/utility tradeoffs. We consider the joint constraints between fairness and robustness, and explore personalized FL to provably address both of them simultaneously. Finally, we study new differentially private optimization methods with improved convergence behavior, achieving state-of-the-art performance under privacy constraints.

Although our work is grounded by the application of federated learning, we show that many of the techniques and fundamental tradeoffs extend well beyond this use-case to more general applications of large-scale and trustworthy machine learning.

Acknowledgments

First and foremost, I am deeply thankful to my awesome advisor, Virginia Smith, for her invaluable guidance and support during my PhD study. She guided me through every aspect of research with great patience, enthusiasm, and vision. In addition to many other things, I learnt from her how to pick the right problems to work on and how to share our research findings with the broader community with both rigor and elegance, which I believe will benefit my entire career. I am also grateful to Ameet Talwalkar for mentoring me on research, serving on my thesis committee, and offering constant encouragement and advice. Additionally, thanks to other committee members, Tianqi Chen, Brendan McMahan, and Dawn Song, for their constructive feedback on my thesis work.

I am extremely fortunate to collaborate with a group of remarkable colleagues and researchers at different institutions, including but not limited to Ahmad Beirami, Jeffrey Bilmes, Yae Jee Cho, Don Dennis, Nageen Himayat, Shengyuan Hu, Gauri Joshi, Alan Liu, Brendan McMahan, Mikhail Khodak, Sashank Reddi, Anit Kumar Sahu, Maziar Sanjabi, Vyas Sekar, Ameet Talwalkar, Tianlong Yu, Manzil Zaheer, and Tianyi Zhou. My special appreciation goes to Ahmad and Manzil for their expertise, taste, and great help outside our research projects. I would also like to call out Manzil for being a great friend and coding guru. I am grateful to gain more insight into practical challenges in industry through my internship at Google Research and the collaboration with researchers there, such as Shanshan Wu, Zachery Charles, Sewoong Oh, and Brendan.

Thanks to Tianqi and Ameet for helping with writing skills requirement, and Zhihao, Aditi, and Chun Kai for evaluating my speaking skills talk.

I would like to extend my thanks to my undergraduate mentors Bin Cui, Ji Liu, Wentao Wu, and Ce Zhang. They showed me how interesting research can be.

My PhD life wouldn't have been so enjoyable and fulfilling without the help from my labmates, friends within and outside CMU, and many researchers I had the opportunity to interact with. I will not list their names here; otherwise, the acknowledgment would be ridiculously long. The COVID pandemic stole almost two years of in-person activities from all of us. Fortunately, I received constant inspiration and support from collaborators, friends, and family, that helped me go through the tough times when I was stuck in a small apartment in Pittsburgh alone.

Finally, I would like to thank my parents for their unconditional love and support. Childhood reading and maths games have had profound impacts on my paths so far. As their only child, I always hope to spend more time with them. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile phones, wearable devices, and smart homes form just a few of the modern distributed networks generating a wealth of data each day. Due to the growing computational power of these devices—coupled with concerns over transmitting private information—it is increasingly attractive to store data locally and push network computation to the edge. For example, *federated learning (FL)* is a learning paradigm that aims to enable efficient and secure data sharing while keeping user data local. FL considers collaboratively training machine learning models across remote devices or siloed organizations in a privacy-preserving manner.

Federated learning has the ability to produce highly accurate statistical models by aggregating knowledge from disparate data sources. Examples of potential applications include: learning sentiment, semantic location, or activities of mobile phone users; adapting to pedestrian behavior in autonomous vehicles; and predicting health events like heart attack risk from wearable devices.

1.1 Canonical Problem Formulation

The canonical federated learning problem involves learning a *single, global* statistical model from data stored on tens to potentially millions of remote devices. We aim to learn this model under the constraint that device-generated data is stored and processed locally, with only intermediate updates being communicated periodically with a central server. In particular, the goal is typically to minimize the following objective function:

$$\min_{w} F(w)\,, \quad \text{where} \quad F(w) := \sum_{k=1}^{N} p_k F_k(w)\,. \tag{1.1}$$

Here, N is the total number of devices, $p_k \geqslant 0$ and $\sum_k p_k = 1$, and F_k is the local objective function for the kth device. The local objective function is often defined as the empirical risk over local data, i.e., $F_k(w) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w; x_{j_k}, y_{j_k})$, where n_k is the number of samples available locally. The user-defined term p_k specifies the relative impact of each device, with two natural settings being $p_k = \frac{1}{n}$ or $p_k = \frac{n_k}{n}$, where $n = \sum_k n_k$ is the

total number of samples. While we will focus on this most popular objective in some chapters, in this thesis, we also study other objectives or modeling approaches that are appropriate depending on the application of interest.

To be successful, the field of federated learning requires fundamental advances in areas such as optimization, systems, and privacy, as federated networks present a variety of practical challenges beyond traditional distributed learning scenarios.

1.2 Challenges of Federated Networks

Federated networks introduce and exacerbate a number of challenges for distributed learning workflows, including expensive communication, systems heterogeneity, and statistical heterogeneity, as discussed in detail in the following.

Challenge 1: Expensive Communication. Communication is a critical bottleneck in federated networks, which, coupled with privacy concerns over sending raw data, necessitates that data generated on each device remain local. Indeed, federated networks are potentially comprised of a massive number of devices, e.g., millions of smart phones, and communication in the network can be slower than local computation by many orders of magnitude [122, 282]. In order to fit a model to data generated by the devices in the federated network, it is therefore necessary to develop communication-efficient methods that iteratively send small messages or *model updates* as part of the training process, as opposed to sending the entire dataset over the network. To further reduce communication in such a setting, two key aspects to consider are: (i) reducing the total number of communication rounds, or (ii) reducing the size of transmitted messages at each round.

Challenge 2: Systems Heterogeneity. The storage, computational, and communication capabilities of each device in federated networks may differ due to variability in hardware (CPU, memory), network connectivity (3G, 4G, 5G, wifi), and power (battery level). Additionally, the network size and systems-related constraints on each device typically result in only a small fraction of the devices being active at once, e.g., hundreds of active devices in a million-device network [36]. Each device may also be unreliable, and it is not uncommon for an active device to drop out at a given iteration due to connectivity or energy constraints. These system-level characteristics dramatically exacerbate challenges such as straggler mitigation and fault tolerance. Federated learning methods that are developed and analyzed must therefore: (i) anticipate a low amount of participation, (ii) tolerate heterogeneous hardware, and (iii) be robust to dropped devices in the network.

Challenge 3: Statistical Heterogeneity. Devices frequently generate and collect data in a non-identically distributed manner across the network, e.g., mobile phone users have varied use of language in the context of a next word prediction task. Moreover, the number of data points across devices may vary significantly, and there may be an underlying

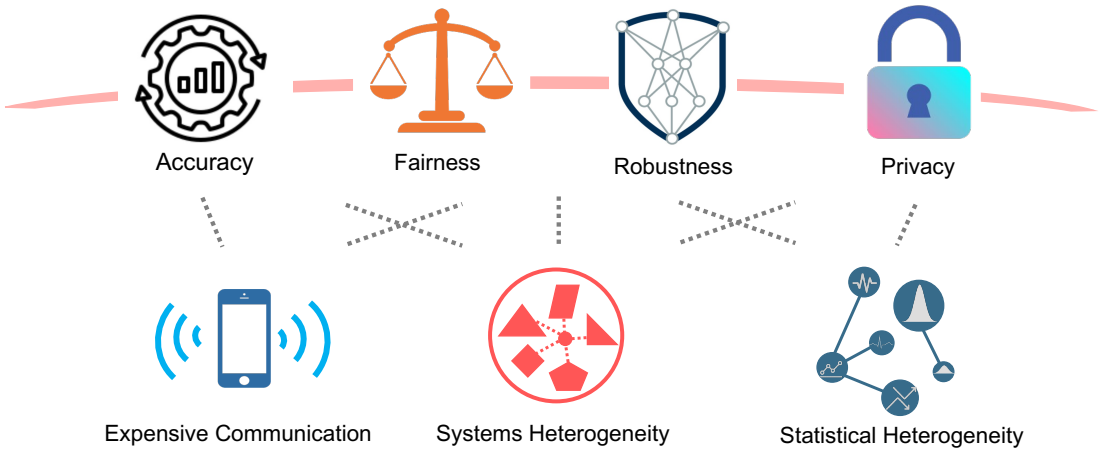


Figure 1.1: Federated networks need to satisfy critical constraints including accuracy/efficiency, fairness, robustness, and privacy. Practical systems and heterogeneity challenges can exacerbate the issues of accuracy and trustworthiness relative to centralized training. The overarching goal of this thesis is to develop principled approaches to understanding and addressing these constraints and their interplays. Taking into account practical challenges of the network, we explore how heterogeneity affects these constraints, and how they interact with one another.

structure present that captures the relationship amongst devices and their associated distributions. This data generation paradigm violates frequently-used independent and identically distributed (I.I.D.) assumptions in distributed optimization, increases the likelihood of stragglers, and may add complexity in terms of modeling, analysis, and evaluation. Indeed, although the canonical federated learning problem of (1.1) aims to learn a single global model, there exist other alternatives such as simultaneously learning distinct local models via multi-task learning frameworks [cf. 264]. There is also a close connection in this regard between leading approaches for federated learning and meta-learning [170]. Enabling *personalized* or *device-specific* modeling is often a more natural approach to handling the statistical heterogeneity of the data.

Finally, privacy is often a major concern in federated learning applications. Federated learning makes a step towards protecting data generated on each device by sharing model updates, e.g., gradient information, instead of the raw data [79, 84]. However, communicating model updates throughout the training process can nonetheless reveal sensitive information [210]. While recent methods aim to enhance the privacy of federated learning using tools such as secure multiparty computation or differential privacy, these approaches often provide privacy at the cost of reduced model performance or system efficiency. Understanding and balancing these tradeoffs, both theoretically and empirically, is a considerable challenge in realizing private federated learning systems.

While federated learning has shown great promise for enabling edge applications, partly as a result of the major challenges above, current FL systems are hindered by

several conflicting constraints. **In this thesis, our goal is to develop principled and scalable approaches to understand and address the challenges of federated networks.** In particular, we aim to study how heterogeneity lies at the center of these practical constraints—not only affecting the final accuracy of the models, but also competing with issues of trustworthiness (fairness, robustness, and privacy), as detailed below.

1.3 Constraints of Federated Learning

Accuracy and Scalability. To be practically useful, federated methods must first and foremost deliver reasonably accurate solutions in an efficient manner, even in light of underlying data and systems heterogeneity.

Trustworthiness. As FL applications aim to learn over client data while using client devices as a distributed computing substrate, there are a number of issues related to trustworthiness that are also critical to consider. These constraints commonly include (a) client **privacy** due to the potential leakage of sensitive information when transmitting information (e.g., model parameters) across the network, (b) **robustness** to device failures, real-world data noise, and adversarial attacks, and (c) delivering **fair** performance to all clients/devices.

Mitigating the potential negative impacts of heterogeneity on federated learning in terms of separate metrics is already challenging, and it is even more so if we jointly consider all metrics which may be fundamentally at odds with each other. For instance, current methods that focus solely on accuracy or fairness are also at odds with robustness constraints, as implementing current approaches can render federated learning systems highly susceptible to adversarial attacks. It is therefore critical to develop and analyze approaches to holistically addressing these connected constraints in realistic networks.

This thesis includes my PhD research on scalable and trustworthy (fair, robust, and private) optimization algorithms and objective functions for both federated learning and general machine learning (Figure 1.1). We rigorously analyze the convergence of our optimization methods, and fairness, robustness, or privacy properties of the proposed approaches. Through evaluation on a suite of realistic federated benchmarks, we demonstrate the practical improvements and efficiency of our works.

1.4 Overview of Contributions

In this section, we summarize the main contributions of this thesis on scalable and trustworthy learning in heterogeneous networks. We discuss background and preliminaries in Chapter 2, and present each work in detail in Chapters 3-7. Although most of the works are grounded by the application of federated learning, we show that many of the techniques and fundamental tradeoffs explored in this thesis extend beyond this use-case to general machine learning applications.

(Chapter 3) ***Accuracy/Scalability:*** **Federated Optimization Methods under Systems and Data Heterogeneity.** FedAvg is the de facto optimization method in the federated setting that allows for local updating and low participation. However, it does not fully address the underlying challenges associated with heterogeneity. We propose `FedProx` [176], which can be viewed as a generalization and re-parametrization of FedAvg (where we add a proximal term to the local objective). While the algorithmic modification is minor, theoretically, we provide the first convergence guarantees for federated optimization methods under realistic heterogeneity assumptions. Practically, we demonstrate that in highly heterogeneous settings, `FedProx` yields more robust convergence. Federated optimization has become an important topic that has received a significant amount of recent research attention [290]. `FedProx` (as well as another principled variant we develop named `FedDane` [175]) takes a first step towards analyzing and improving federated optimization under practical constraints (device unreliability, expensive communication, and heterogeneity).

(Chapter 4) ***Fairness:*** **Addressing Representation Disparity.** Heterogeneity not only affects the convergence of federated learning methods, but also poses challenges to the *fairness* of the final model performance. For example, while models trained via federated learning may be accurate on average, these models can sometimes perform unfairly or even catastrophically on subsets of the network. This type of unfairness is known as *representation disparity*, and is a critical concern especially in massive cross-device federated settings. To address unfairness stemming from heterogeneity, we developed q-Fair FL (q-FFL) [182], a new optimization objective to minimize an aggregate *reweighted* loss parameterized by q such that the devices with higher loss are given higher relative weight. This framework allows for a *tunable* fairness/utility tradeoff through the hyperparameter q, and is easy to implement in a scalable fashion.

(Chapter 5) ***Robustness:*** **Handling Competing Constraints Between Fairness and Robustness Through Personalization.** Simultaneously satisfying the constraints of accuracy, fairness, and robustness is exceptionally difficult. For instance, current fairness approaches usually upweight worst-performing devices, thus being highly susceptible to training time attacks from malicious devices. On the other hand, robust methods may filter out rare but informative updates, causing unfairness. In this setting, *heterogeneity* is again a root cause for tension between these constraints—and is key in paving a path forward. With this insight, we propose a multi-task learning (a framework that learns shared, heterogeneous models) method, `Ditto`, as a unified approach for satisfying the diverse constraints in federated learning [186]. Theoretically, we analyze the ability of `Ditto` to achieve fairness and robustness simultaneously on a class of linear problems. Empirically, we demonstrate that `Ditto` not only achieves competitive performance relative to recent personalization methods, but also enables more accurate, robust, and fair models relative to state-of-the-art fair or robust baselines. Our `Ditto` work opens up many interesting questions regarding the connections between personalization and practical constraints.

(Chapter 6) ***Privacy:*** **Private Adaptive Optimization.** Privacy is a critical concern in both centralized and federated learning, but the study of differential privacy (DP) in machine learning has been largely limited to (variants) of the very basic SGD optimizer. Considering the widely-used adaptive methods (such as AdaGrad), we observed that privacy costs may negate the benefits of adapting to gradient geometry. In particular, the preconditioners might become less effective (e.g., being dominated by DP noise) if naively privatizing existing adaptive optimizers. We proposed to leverage some structures of gradients to construct more effective preconditioners with reduced privacy noise, where the resulting algorithms can be applied to federated settings as well. For instance, we leverage non-sensitive side information to precondition the gradients [187]. In other cases where such information is not available, we rely on the observation that adaptive methods can tolerate stale preconditioners, and constructed delayed but less noisy preconditioners from historical gradients [188]. We prove that in both cases, the private adaptive methods can reduce the amount of privacy noise when the gradients are sparse. This takes an initial step towards guaranteeing differential privacy by default in conjunction with modern learning objectives and algorithms.

(Chapter 7) ***Extensions to General ML Problems:*** **Tilted Empirical Risk Minimization** The insights from q-FFL can be further extended to general ML problems. In particular, the empirical risk minimization objective is typically designed to perform well on the average loss, which can result in models that are sensitive to outliers, generalize poorly, or treat subgroups unfairly. We propose tilted empirical risk minimization (TERM) [184, 185], a unified framework to address all these issues through a flexible parameter called the tilt. By tuning this parameter, we show that we are able to tune the impact of individual losses to provably decrease the influence of outliers, enable fairness or robustness, and provide better generalization. We additionally analyze various properties of TERM and its connections with other risk-averse formulations and apply it to numerous modern ML problems. TERM can match the performance of state-of-the-art approaches on many applications, converges almost as quickly as standard mini-batch SGD, and has inspired the design of new objectives with favorable properties. Both the q-FFL and TERM objectives have become common baselines in the areas of fair/robust learning.

1.5 Excluded Work

Federated learning differs significantly from traditional distributed environments—requiring fundamental advances in areas such as privacy, large-scale machine learning, and distributed optimization, and raising new questions at the intersection of diverse fields, such as machine learning and systems [177]. In this section, I describe the research I am involved in that is not included as part of this thesis.

To address the entire pipeline, it is important to develop comprehensive solutions considering various critical aspects of real-world scenarios including *system automation, tools and benchmarks, and incorporation of domain knowledge*. Other projects on this front

(where I am not the primary contributor) include hyperparameter tuning at scale in federated settings [154], better client selection strategies in massive networks [24], lifecycle management of machine learning systems and operations, applying federated learning to the application of anomaly detection across IoT devices [313], and incentivizing clients to participate in federation [53].

We have also developed common resources for federated learning that have been widely adopted by the community, including an FL survey [177] and two benchmarks on federated datasets [43][1] and personalized federated learning [301] to help with reproducibility and facilitate research progress in the related areas.

1.6 Bibliographic Notes

The research presented in this thesis is based on joint work with many co-authors, as described below. In each work, I am either the primary contributor or one of two equal primary contributors.

Chapter 3 is based on joint work with Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith [175, 176]. Chapter 4 and Chapter 7 are joint works with Ahmad Beirami, Maziar Sanjabi, and Virginia Smith [182, 184, 185]. Chapter 5 is based on joint work with Shengyuan Hu, Ahmad Beirami, and Virginia Smith [186]. Chapter 6 is based on joint works with Manzil Zaheer, Ken Ziyu Liu, Sashank Reddi, Brendan McMahan, and Virginia Smith [187, 188].

[1] leaf.cmu.edu

Chapter 2

Preliminaries and Related Work

In this chapter, we discuss the background and related work of this thesis. As federated learning (or machine learning) is a quickly growing area, we do not aim to cover complete literature of each topic; but rather, discuss only closely-related works, highlight their connections with our works, and defer interested readers to other references. We also set up definitions of fairness, robustness, and privacy considered in this thesis.

2.1 Distributed and Federated Optimization

Large-scale machine learning, particularly in data center settings, has motivated the development of numerous distributed optimization methods in the past decade [see, e.g., 39, 65, 66, 172, 236, 241, 253, 267, 323, 325]. However, as computing substrates such as phones, sensors, and wearable devices grow both in power and in popularity, it is increasingly attractive to learn statistical models locally in networks of distributed devices, in contrast to moving the data to the data center. Recent optimization methods have been proposed that are tailored to the specific challenges in the federated setting. These methods have shown significant improvements over traditional distributed approaches such as ADMM [39] or mini-batch methods [66] by allowing both for inexact local updating in order to balance communication vs. computation in large networks, and for a small subset of devices to be active at any communication round [208, 264]. For example, Smith et al. [264] propose a communication-efficient primal-dual optimization method that learns separate but related models for each device through a multi-task learning framework. In the non-convex setting, Federated Averaging (FedAvg), a heuristic method based on averaging local Stochastic Gradient Descent (SGD) updates in the primal, has instead been shown to work well empirically [208].

Unfortunately, FedAvg is quite challenging to analyze due to its local updating scheme, the fact that few devices are active at each round, and the issue that data is frequently distributed in a heterogeneous nature in the network. In particular, as each device generates its own local data, *statistical heterogeneity* is common with data being non-identically distributed between devices. Several works have made steps towards analyzing FedAvg in simpler, non-federated settings. For instance, parallel SGD and

related variants [191, 236, 253, 272, 289, 300, 323, 329], which make local updates similar to FedAvg, have been studied in the IID setting. However, the results rely on the premise that each local solver is a copy of the same stochastic process (due to the IID assumption). This line of reasoning does not apply to the heterogeneous setting.

Although some recent works [110, 131, 294, 311] have explored convergence guarantees in statistically heterogeneous settings, they make the limiting assumption that all devices participate in each round of communication, which is often infeasible in realistic federated networks [208]. Further, they rely on specific solvers to be used on each device (either SGD or GD), as compared to the solver-agnostic framework proposed herein, and add additional assumptions of convexity [294] or uniformly bounded gradients [311] to their analyses. There are also heuristic approaches that aim to tackle statistical heterogeneity by sharing the local device data or server-side proxy data [123, 127, 327]. However, these methods may be unrealistic: in addition to imposing burdens on network bandwidth, sending local data to the server [127] violates the key privacy assumption of federated learning, and sending globally-shared proxy data to all devices [123, 327] requires effort to carefully generate or collect such auxiliary data.

Beyond statistical heterogeneity, *systems heterogeneity* is also a critical concern in federated networks. The storage, computational, and communication capabilities of each device in federated networks may differ due to variability in hardware (CPU, memory), network connectivity (3G, 4G, 5G, wifi), and power (battery level). These system-level characteristics dramatically exacerbate challenges such as straggler mitigation and fault tolerance. One strategy used in practice is to ignore the more constrained devices failing to complete a certain amount of training [36]. However, this can have negative effects on convergence as it limits the number of effective devices contributing to training, and may induce bias in the device sampling procedure if the dropped devices have specific data characteristics.

Our `FedProx` framework (Chapter 3) is capable of handling heterogeneous federated environments while maintaining similar privacy and computational benefits. We analyze the convergence behavior of the framework through a *statistical* dissimilarity characterization between local functions, while also taking into account practical *systems* constraints. Our dissimilarity characterization is inspired by the randomized Kaczmarz method for solving linear system of equations [139, 274], a similar assumption of which has been used to analyze variants of SGD in other settings [see, e.g., 250, 283, 310]. Our proposed framework provides improved robustness and stability for optimization in heterogeneous federated networks.

2.2 Fair Federated Learning

Fairness in Machine Learning. Fairness is a broad topic that has received much attention in the machine learning community, though the goals often differ from that described in this work. Indeed, *fairness* in machine learning is typically defined as the protection of some specific attribute(s). Two common approaches are to preprocess the data to remove information about the protected attribute, or to post-process the model

by adjusting the prediction threshold after classifiers are trained [44, 91, 111]. Another set of works optimize an objective subject to some fairness constraints during training time [7, 23, 59, 83, 113, 299, 315, 316]. Our fairness work described in Chapter 4 also enforces fairness during training, although we define fairness as the uniformity of the accuracy distribution across devices (Definition 1) in federated learning, as opposed to the protection of a specific attribute. Although some works define accuracy parity to enforce equal error rates among specific groups as a notion of fairness [59, 315], devices in federated networks may not be partitioned by protected attributes, and our goal is not to optimize for identical accuracy across all devices. Cotter et al. [59] use a notion of 'minimum accuracy', which is conceptually similar to our goal. However, it requires one optimization constraint for each device, which would result in hundreds to millions of constraints in federated networks.

Fairness in Resource Allocation. Fair resource allocation has been extensively studied in fields such as network management [85, 106, 151, 221] and wireless communications [86, 220, 248, 256]. In these contexts, the problem is defined as allocating a scarce shared resource, e.g., communication time or power, among many users. In these cases, directly maximizing utilities such as total throughput may lead to unfair allocations where some users receive poor service. As a service provider, it is important to improve the quality of service for all users while maintaining overall throughput. For this reason, several popular fairness measurements have been proposed to balance between fairness and total throughput, including Jain's index [126], entropy [239], max-min/min-max fairness [234], and proportional fairness [150]. A unified framework is captured through α-fairness [164, 214], in which the network manager can tune the emphasis on fairness by changing a single parameter, α.

To draw an analogy between federated learning and the problem of resource allocation, one can think of the global model as a resource that is meant to serve the users (or devices). In this sense, it is natural to ask similar questions about the fairness of the service that users receive and use similar tools to promote fairness. Despite this, we are unaware of any works that use α-fairness from resource allocation to modify objectives in machine learning. Inspired by the α-fairness metric, we propose a similarly modified objective, q-Fair Federated Learning (q-FFL) (Chapter 4), to encourage a more fair accuracy distribution across devices in the context of federated training (Definition 1). Similar to the α-fairness metric, our q-FFL objective is flexible enough to enable tradeoffs between fairness and other traditional metrics such as accuracy by changing the parameter q. In Section 4.3, we show empirically that the use of q-FFL as an objective in federated learning enables a more uniform accuracy distribution across devices—significantly reducing variance while maintaining the average accuracy.

Fairness in Federated Learning. Due to the heterogeneity of the data in federated networks, it is possible that the performance of a model will vary significantly across the devices. This concern, also known as *representation disparity* [113], is a major challenge in FL, as it can potentially result in uneven outcomes for the devices. Throughout the thesis,

our fairness notion in the context of federated learning is formally defined as below.

Definition 1 (Fairness)**.** *We say that a model w_1 is more fair than w_2 if the test performance distribution of w_1 across the network is* more uniform *than that of w_2, i.e.,* std $\{F_k(w_1)\}_{k\in[K]} <$ std $\{F_k(w_2)\}_{k\in[K]}$ *where $F_k(\cdot)$ denotes the test loss on device $k\in[K]$, and* std$\{\cdot\}$ *denotes the standard deviation. In the presence of adversaries, we measure fairness only on benign devices.*

We note that there exists a tension between variance and utility in the definition above; in general, a common goal is to lower the variance while maintaining a reasonable average performance (e.g., average test accuracy). To address representation disparity, it is common to use minimax optimization [68, 215] or flexible sample reweighting approaches [182, 184] to encourage a more uniform quality of service. For instance, Mohri et al. [215] proposed a minimax optimization scheme, Agnostic Federated Learning (`AFL`), which optimizes for the performance of the single worst device. This method has only been applied at small scales (for a handful of devices). Compared to `AFL`, our fairness objective (Chapater 4) is more flexible as it can be tuned based on the desired amount of fairness. In all cases, by up-weighting the importance of rare devices or data, fair methods may not be robust in that they can easily overfit to corrupted devices. The tension between fairness and robustness (defined broadly) has been studied in previous works, though for different notions of fairness (equalized odds) or robustness (backdoor attacks) [288], or in centralized settings [47]. In Chapter 5, we explore two specific forms of fairness (defined herein) and robustness (see next section), and develop new methods to address them jointly.

2.3 Robust Federated Learning

Training-time attacks (including data poisoning and model poisoning) have been extensively studied in prior work [33, 52, 80, 103, 124, 124, 194, 252, 288, 303]. In federated settings, a number of strong attack methods have been explored, including scaling malicious model updates [22], collaborative attacking [276], defense-aware attacks [29, 90], and adding edge-case adversarial training samples [288].

In Chapter 5, we investigate a simple, scalable technique to simultaneously improve accuracy, fairness, and robustness in federated learning. Our work aims to investigate common attacks related to Byzantine robustness [163], as formally described below.

Definition 2 (Robustness)**.** *We are conceptually interested in Byzantine robustness [163], where the malicious devices can send arbitrary updates to the server to compromise training. To measure robustness, we assess the mean test performance on benign devices, i.e., we consider model w_1 to be more robust than w_2 to a specific attack if the mean test performance across the benign devices is higher for model w_1 than w_2 after training with the attack. We examine three widely-used attacks in our threat model:*

- *(A1) Label poisoning*: Corrupted devices do not have access to the training APIs and training samples are poisoned with flipped (if binary) or uniformly random noisy

labels [29, 32].

- *(A2) Random updates*: Malicious devices send random zero-mean Gaussian parameters [304].
- *(A3) Model replacement*: Malicious devices scale their adversarial updates to make them dominate the aggregate updates [22].

While non-exhaustive, these attacks have been commonly studied in distributed and federated settings, and explore corruption at various points (the underlying data, labels, or model). In terms of defenses, robust aggregation is a common strategy to mitigate the effect of malicious updates [35, 114, 171, 230, 277]. Other defenses include gradient clipping [277] or normalization [121]. While these strategies can improve robustness, they may also produce *unfair* models by filtering out informative updates, especially in heterogeneous settings [288]. In our experiments, we compare our proposed approach with several strong defenses (median, gradient clipping [277], Krum, Multi-Krum [35], gradient-norm based anomaly detector [22], and a new defense proposed herein) and show that our approach can improve both robustness and fairness compared with these methods.

2.4 Personalized Federated Learning

Given the variability of data in federated networks, personalization is a natural approach used to improve accuracy. Numerous works have proposed techniques for personalized federated learning. Smith et al. [265] first explore personalized FL via a primal-dual MTL framework, which applies to convex settings. Personalized FL has also been explored through clustering [e.g., 100, 216, 249], finetuning/transfer learning [312, 328], meta-learning [50, 88, 132, 153, 169, 261], and other forms of MTL, such as hard model parameter sharing [8, 190] or the weighted combination method in Zhang et al. [322]. Our work (Chapter 5) differs from these approaches by simultaneously learning local and global models via a global-regularized MTL framework, which applies to non-convex ML objectives.

Similar in spirit to our approach are works that interpolate between global and local models [69, 202]. However, as discussed in Deng et al. [69], these approaches can effectively reduce to local minimizers without additional constraints. The most closely related works are those that regularize personalized models towards their average [72, 107, 108], which can be seen as a form of classical mean-regularized MTL [87]. Our objective is similarly inspired by mean-regularized MTL, although we regularize towards a global model rather than the average personalized model. One advantage of this is that it allows for methods designed for the global federated learning problem (e.g., optimization methods, privacy/security mechanisms) to be easily re-used in our framework, with the benefit of additional personalization. We compare against a range of personalized methods empirically, showing that our approach achieves similar or superior performance across a number of common FL benchmarks.

Finally, a key contribution of our work is jointly exploring the robustness and fairness

benefits of personalized FL. The benefits of personalization for fairness alone have been demonstrated empirically in prior work [109, 291]. Connections between personalization and robustness have also been explored in Yu et al. [312], although the authors propose using personalization methods on top of robust mechanisms. Our work differs from these works by arguing that MTL itself offers inherent robustness and fairness benefits, and exploring the challenges that exist when attempting to satisfy both constraints simultaneously.

2.5 Differentially Private Machine Learning

Differentially private (DP) training is a common tool for federated learning, which inspires our work; but our research on private optimization applies to general ML problems (Chapter 6). Informally, differential privacy in machine learning offers protection by masking the influence of individual examples (example-level DP, e.g. [4, 25, 269]) or all of the examples from one user (user-level DP, e.g. [143, 210]) on the trained model. There are many algorithms to achieve this, including object perturbation, gradient perturbation, and model perturbation. This thesis focuses on the popular gradient perturbation method with Gaussian mechanisms [84]. Without additional assumptions on the problem structure, DP algorithms can suffer from $O(\frac{\sqrt{d}}{n\varepsilon})$ excess empirical risk where d is the dimension of the model parameters and n is the number of training samples [25]. It is possible to mitigate such a dependence in the unconstrained setting [141, 270] or assuming oracle access to a constant-rank gradient subspace [141, 333]. However, these assumptions may not be satisfied in practice.

Unless much larger batch sizes and possibly larger datasets are used, DP mechanisms often lead to a significant utility drop. Extensive research has thus been devoted to investigating improved privacy/utility/computation tradeoffs for DP-SGD, including various training techniques (e.g., data augmentation and large-batch training) [64], leveraging public data [15, 333], and releasing gradient statistics via tree aggregation to reduce the amount of noise [46, 70, 143]. These prior works are orthogonal to and could be applied in conjunction with our proposed method in Chapter 6, focusing on adaptive optimizers. Recent work [15] proposes to use public data differently (evaluating public loss as the mirror map in a mirror descent algorithm) to obtain dimension-independent bounds. However, this method do not account for gradient preconditioning, as their approximation is a linear combination of private and public gradients.

In gradient-based optimization, adaptive optimizers and their properties have been extensively studied [e.g., 78, 155, 218, 318]. They effectively result in coordinate-wise learning rates, which can be advantageous for many learning problems. The preconditioners can be estimated via a moving average of mini-batch gradients (as in, e.g., Adam) or simply by calculating the sum of gradients so far (AdaGrad). In the context of adaptive differentially private optimization, we note that 'adaptivity' may have various meanings. For example, Andrew et al. [16] adaptively set the clipping threshold based on the private estimation of gradient norms. To reduce privacy cost in iterative DP algorithms, it is natural to consider applying adaptive optimizers (e.g., AdaGrad [78, 207],

RMSProp [117], AMSGrad [237], and Yogi [318]) to speed up convergence. Our work is related to this line of work that aims to develop and analyze differentially private variants of common adaptive optimizers (e.g., private AdaGrad, private Adam), which mostly focus on estimating gradient statistics from noisy gradients [20, 229, 332]. However, estimating gradient moments in this way may yield preconditioners with too much noise, resulting in adaptive methods that may not have meaningful improvements over DP-SGD.

In Chapter 6, we propose two methods to handle this issue. The first one (named `AdaDPS`) leverages side information (such as summary statistics or proxy data) to estimate the preconditioner. Previous works differ from `AdaDPS` in the order of preconditioning and privatization, the techniques used to approximate the gradient geometry, and the convergence analysis (see Section 6.3 for details). In addition, without access to public data, we propose `DP`^{2}, using delayed preconditioners to reduce noise. We also note that previous works have explored the high-level direction of delayed preconditioners, but mainly as a compromise for computational considerations in non-private training [104]. In `DP`^{2}, we instead show that staleness can be leveraged to improve privacy/utility tradeoffs in private adaptive optimization.

In terms of privacy formulations, we consider classic sample-level DP in centralized settings, and a variant of it—*user-level DP*—in distributed/federated environments. We define both more formally below.

Definition 3 (Differential privacy [82])**.** *A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for all neighbouring datasets D, D' differing by one element, and every possible subset of outputs O,*

$$\Pr(\mathcal{M}(D) \in O) \leqslant e^{\varepsilon} \Pr\big(\mathcal{M}(D') \in O\big) + \delta.$$

Within DP, neighbouring datasets can be defined in different ways depending on the application of interest. In this work, we also apply `AdaDPS` to federated learning, where differential privacy is commonly defined at the granularity of users/devices [144, 210], as stated below.

Definition 4 (User-level DP for federated learning [210])**.** *A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for all datasets U, U' differing by one user, and every possible subset of outputs O,*

$$\Pr(\mathcal{M}(U) \in O) \leqslant e^{\varepsilon} \Pr\big(\mathcal{M}(U') \in O\big) + \delta.$$

2.6 Fair or Robust Learning Beyond FL

Our insights in the fair federated learning work [182] lead us to develop and analyze a unified objective function—tilted empirical risk minimization (TERM)—to handle fairness and robustness. TERM is parameterized by a constant t to control the impact of individual losses. Our TERM work [184, 185] (Chapter 7) applies broadly to a range

of ML applications, achieving competitive performance with state-of-the-art, problem-specific approaches. Here we discuss related problem-specific works in machine learning addressing deficiencies of ERM to achieve fairness or robustness and their connections with TERM. We roughly group them into alternate aggregation schemes, alternate loss functions, and sample re-weighting schemes.

Alternate Aggregation Schemes. A common alternative to the standard average loss in empirical risk minimization is to consider a min-max objective, which aims to minimize the max-loss. Min-max objectives are commonplace in machine learning, and have been used for a wide range of applications, such as ensuring fairness across subgroups [113, 161, 215, 247, 271, 281], enabling robustness under small perturbations [262], or generalizing to unseen domains [285]. Our TERM objective can be viewed as a minimax smoothing [157, 227] with the added flexibility of a tunable t to allow the user to optimize utility for different quantiles of loss similar to superquantile approaches [160, 243], directly trading off between robustness/fairness and utility for positive and negative values of t. However, the TERM objective remains smooth (and efficiently solvable) for moderate values of t, resulting in faster convergence even when the resulting solutions are effectively the same as the min-max solution or other desired quantiles of the loss. Interestingly, Cohen et al. introduce Simnets [55, 56], with a similar exponential smoothing operator, though for a differing purpose of achieving layer-wise operations *between* sum and max in deep neural networks.

Alternate Loss Functions. Rather than modifying the way the losses are aggregated, as in (smoothed) min-max or superquantile methods, it is also quite common to modify the losses themselves. For example, in robust regression, it is common to consider losses such as the L_1 loss, Huber loss, or general M-estimators [119] as a way to mitigate the effect of outliers [30]. Wang et al. [295] study a similar exponentially tilted loss for robust regression and characterize the break down point, though it is limited to the squared loss and only corresponds to TERM with $t<0$. Losses can also be modified to address outliers by favoring small losses [314, 326] or gradient clipping [211]. Some works mitigate label noise by explicitly modeling noise distributions into end-to-end training combined with an additional noise model regularizer [135, 136]. On the other extreme, the largest losses can be magnified to encourage focus on hard samples [182, 192, 296], which is a popular approach for curriculum learning. Constraints could also be imposed to promote fairness during the optimization procedure [13, 23, 58, 73, 111, 198, 231, 240, 315, 317]. A line of work proposes α-loss, which is able to promote fairness or robustness for classification tasks [278]. Ignoring the log portion of the objective, TERM can be viewed as an alternate loss function exponentially shaping the loss to achieve both of these goals with a single objective, i.e., magnifying hard examples with $t > 0$ and suppressing outliers with $t < 0$. In addition, we show that TERM can even achieve both goals simultaneously with hierarchical multi-objective optimization.

Sample Re-Weighting Schemes. There exist approaches that implicitly modify the underlying ERM objective by re-weighting the influence of the samples themselves. These re-weighting schemes can be enforced in many ways. A simple and widely used example is to subsample training points in different classes. Alternatively, one can re-weight examples according to their loss function when using a stochastic optimizer, which can be used to put more emphasis on "hard" or "unfair" examples [6, 128, 149, 166, 257]. Re-weighting can also be implicitly enforced via the inclusion of a regularization parameter [5], loss clipping [306], or modelling crowd-worker qualities [152]. Such an explicit re-weighting has been explored for other applications [e.g., 48, 96, 130, 192, 238, 258], though in contrast to these methods, TERM is applicable to a general class of loss functions, with theoretical guarantees. TERM is equivalent to a dynamic re-weighting of the samples based on the values of the objectives, which could be viewed as a convexified version of loss clipping. We note that such view holds more generally for all distributionally robust objectives [263]. We compare to several sample re-weighting schemes empirically.

Chapter 3

Accuracy: Federated Optimization in Heterogeneous Networks

The first question we would like to ask in this thesis is: How does systems and statistical heterogeneity affect optimization? How can we develop methods to ensure that federated methods converge to the correct solution efficiently under heterogeneity? This chapter presents a scalable and principled optimization framework to explore thesis questions, leveraging a device similarity characterization.

3.1 Overview

Federated learning has emerged as an attractive paradigm for distributing training of machine learning models in networks of remote devices. While there is a wealth of work on distributed optimization in the context of machine learning, two key challenges distinguish federated learning from traditional distributed optimization: high degrees of *systems and statistical heterogeneity*[1] [178, 208].

In an attempt to handle heterogeneity and tackle high communication costs, optimization methods that allow for local updating and low participation are a popular approach for federated learning [208, 264]. In particular, `FedAvg` [208] is an iterative method that has emerged as the de facto optimization method in the federated setting. At each iteration, `FedAvg` first locally performs E epochs of stochastic gradient descent (SGD) on K devices—where E is a small constant and K is a small fraction of the total devices in the network. The devices then communicate their model updates to a central server, where they are averaged.

While `FedAvg` has demonstrated empirical success in heterogeneous settings, it does not fully address the underlying challenges associated with heterogeneity. In the context of systems heterogeneity, `FedAvg` does not allow participating devices to perform vari-

[1] Privacy is a third key challenge in the federated setting. While not the focus of this work, standard privacy-preserving approaches such as differential privacy and secure multiparty communication can naturally be combined with the methods proposed herein—particularly since our framework proposes only lightweight algorithmic modifications to prior work.

able amounts of local work based on their underlying systems constraints; instead it is common to simply drop devices that fail to compute E epochs within a specified time window [36]. From a statistical perspective, `FedAvg` has been shown to diverge empirically in settings where the data is non-identically distributed across devices [e.g., 208, Sec 3]. Unfortunately, `FedAvg` is difficult to analyze theoretically in such realistic scenarios and thus lacks convergence guarantees to characterize its behavior (see Section 2.1 for additional details).

In this work, we propose `FedProx`, a federated optimization algorithm that addresses the challenges of heterogeneity both theoretically and empirically. A key insight we have in developing `FedProx` is that an interplay exists between systems and statistical heterogeneity in federated learning. Indeed, both dropping stragglers (as in `FedAvg`) or naively incorporating partial information from stragglers (as in `FedProx` with the proximal term set to 0) implicitly increases statistical heterogeneity and can adversely impact convergence behavior. To mitigate this issue, we propose adding a proximal term to the objective that helps to improve the stability of the method. This term provides a principled way for the server to account for heterogeneity associated with partial information. Theoretically, these modifications allow us to provide convergence guarantees for our method and to analyze the effect of heterogeneity. Empirically, we demonstrate that the modifications improve the stability and overall accuracy of federated learning in heterogeneous networks—improving the absolute testing accuracy by 22% on average in highly heterogeneous settings.

We present our proposed framework, `FedProx`, in Section 3.2, and derive convergence guarantees for the framework accounting for both statistical and systems heterogeneity in Section 3.3. Finally, in Section 3.4, we provide a thorough empirical evaluation of `FedProx` on a suite of synthetic and real-world federated datasets. Our empirical results help to illustrate and validate our theoretical analysis, and demonstrate the practical improvements of `FedProx` over `FedAvg` in heterogeneous networks.

3.2 Federated Optimization: Methods

In this section, we introduce the key ingredients behind recent methods for federated learning, including `FedAvg`, and then outline our proposed framework, `FedProx`.

Federated learning methods [e.g., 208, 264] are designed to handle multiple devices collecting data and a central server coordinating the global learning objective across the network. In particular, as mentioned in Chapter 1, the aim is to minimize:

$$\min_w \; f(w) = \sum_{k=1}^{N} p_k F_k(w) = \mathbb{E}_k[F_k(w)], \tag{3.1}$$

where N is the number of devices, $p_k \geqslant 0$, and $\sum_k p_k$=1. In general, the local objectives measure the local empirical risk over possibly differing data distributions $\mathcal{D}_k$, i.e., $F_k(w) := \mathbb{E}_{x_k \sim \mathcal{D}_k}[f_k(w; x_k)]$, with n_k samples available at each device k. Hence, we can

set $p_k = \frac{n_k}{n}$, where $n = \sum_k n_k$ is the total number of data points. In this work, we consider $F_k(w)$ to be possibly non-convex.

To reduce communication, a common technique in federated optimization is that on each device, a *local objective function* based on the device's data is used as a surrogate for the global objective function. At each outer iteration, a subset of the devices are selected and *local solvers* are used to optimize the local objective functions on each of the selected devices. The devices then communicate their local model updates to the central server, which aggregates them and updates the global model accordingly. The key to allowing flexible performance in this scenario is that each of the local objectives can be solved *inexactly*. This allows the amount of local computation vs. communication to be tuned based on the number of local iterations that are performed (with additional local iterations corresponding to more exact local solutions). We introduce this notion formally below, as it will be utilized throughout the chapter.

Definition 5 (γ-inexact solution)**.** *For a function $h(w; w_0) = F(w) + \frac{\mu}{2}\|w - w_0\|^2$, and $\gamma \in [0, 1]$, we say w^* is a γ-inexact solution of $\min_w h(w; w_0)$ if $\|\nabla h(w^*; w_0)\| \leqslant \gamma \|\nabla h(w_0; w_0)\|$, where $\nabla h(w; w_0) = \nabla F(w) + \mu(w - w_0)$. Note that a smaller γ corresponds to higher accuracy.*

We use γ-inexactness in our analysis (Section 3.3) to measure the amount of local computation from the local solver at each round. As discussed earlier, different devices are likely to make different progress towards solving the local subproblems due to variable systems conditions, and it is therefore important to allow γ to vary both by device and by iteration. This is one of the motivations for our proposed framework discussed in the next sections. For ease of notation, we first derive our main convergence results assuming a uniform γ as defined here (Section 3.3), and then provide results with variable γ's in Corollary 2.

3.2.1 Federated Averaging (`FedAvg`)

In Federated Averaging (`FedAvg`) [208], the local surrogate of the global objective function at device k is $F_k(\cdot)$, and the local solver is stochastic gradient descent (SGD), with the same learning rate and number of local epochs used on each device. At each round, a subset $K \ll N$ of the total devices are selected and run SGD locally for E number of epochs, and then the resulting model updates are averaged. The details of `FedAvg` are summarized in Algorithm 1.

McMahan et al. [208] show empirically that it is crucial to tune the optimization hyperparameters of `FedAvg` properly. In particular, the number of local epochs in `FedAvg` plays an important role in convergence. On one hand, performing more local epochs allows for more local computation and potentially reduced communication, which can greatly improve the overall convergence speed in communication-constrained networks. On the other hand, with dissimilar (heterogeneous) local objectives F_k, a larger number of local epochs may lead each device towards the optima of its local objective as opposed to the global objective—potentially hurting convergence or even causing the method to diverge. Further, in federated networks with heterogeneous systems resources, setting

Algorithm 1 Federated Averaging (`FedAvg`)

Input: K, T, η, E, w^0, N, p_k, $k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each device $k \in S_t$ updates w^t for E epochs of SGD on F_k with step-size η to obtain w_k^{t+1}
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as $w^{t+1} = \frac{1}{K} \sum_{k \in S_t} w_k^{t+1}$
end for

the number of local epochs to be high may increase the risk that devices do not complete training within a given communication round and must therefore drop out of the procedure [36].

In practice, it is therefore important to find a way to set the local epochs to be high (to reduce communication) while also allowing for robust convergence. More fundamentally, we note that the 'best' setting for the number of local epochs is likely to change at each iteration and on each device—as a function of both the local data and available systems resources. Indeed, a more natural approach than mandating a *fixed* number of local epochs is to allow the epochs to *vary* according to the characteristics of the network, and to carefully merge solutions by accounting for this heterogeneity. We formalize this strategy in `FedProx`, introduced below.

3.2.2 Proposed Framework: `FedProx`

Our proposed framework, `FedProx` (Algorithm 2), is similar to `FedAvg` in that a subset of devices are selected at each round, local updates are performed, and these updates are then averaged to form a global update. However, `FedProx` makes the following simple yet critical modifications, which result in significant empirical improvements and also allow us to provide convergence guarantees for the method.

Tolerating Partial Work. As previously discussed, different devices in federated networks often have different resource constraints in terms of the computing hardware, network connections, and battery levels. Therefore, it is unrealistic to force each device to perform a uniform amount of work (i.e., running the same number of local epochs, E), as in `FedAvg`. In `FedProx`, we generalize `FedAvg` by allowing for variable amounts of work to be performed locally across devices based on their available systems resources, and then aggregate the partial solutions sent from the stragglers (as compared to dropping these devices). In other words, instead of assuming a uniform γ for all devices throughout the training process, `FedProx` implicitly accommodates variable γ's for different devices and at different iterations. We formally define γ_k^t-inexactness for device k at iteration t below,

which is a natural extension from Definition 5.

Definition 6 (γ_k^t-inexact solution)**.** *For a function $h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2$, and $\gamma \in [0, 1]$, we say w^* is a γ_k^t-inexact solution of $\min_w h_k(w; w_t)$ if $\|\nabla h_k(w^*; w_t)\| \leqslant \gamma_k^t \|\nabla h_k(w_t; w_t)\|$, where $\nabla h_k(w; w_t) = \nabla F_k(w) + \mu(w - w_t)$. Note that a smaller γ_k^t corresponds to higher accuracy.*

Analogous to Definition 5, γ_k^t measures how much local computation is performed to solve the local subproblem on device k at the t-th round. The variable number of local iterations can be viewed as a proxy of γ_k^t. Utilizing the more flexible γ_k^t-inexactness, we can readily extend the convergence results under Definition 5 (Theorem 1) to consider issues related to systems heterogeneity such as stragglers (see Corollary 2).

Proximal Term. As mentioned in Section 3.2.1, while tolerating nonuniform amounts of work to be performed across devices can help alleviate negative impacts of systems heterogeneity, too many local updates may still (potentially) cause the methods to diverge due to the underlying heterogeneous data. We propose to add a proximal term to the local subproblem to effectively limit the impact of variable local updates. In particular, instead of just minimizing the local function $F_k(\cdot)$, device k uses its local solver of choice to approximately minimize the following objective h_k:

$$\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 . \tag{3.2}$$

The proximal term is beneficial in two aspects: (1) It addresses the issue of statistical heterogeneity by restricting the local updates to be closer to the initial (global) model without any need to manually set the number of local epochs. (2) It allows for safely incorporating variable amounts of local work resulting from systems heterogeneity. We summarize the steps of `FedProx` in Algorithm 2.

Algorithm 2 `FedProx` (Proposed Framework)

1: **Input:** K, T, μ, γ, w^0, N, p_k, $k = 1, \cdots, N$
2: **for** $t = 0, \cdots, T-1$ **do**
3: Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
4: Server sends w^t to all chosen devices
5: Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w\ h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
6: Each device $k \in S_t$ sends w_k^{t+1} back to the server
7: Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
8: **end for**

We note that proximal terms such as the one above are a popular tool utilized throughout the optimization literature; for completeness, we provide a more detailed discussion

on this in Appendix 3.6. An important distinction of the proposed usage is that we suggest, explore, and analyze such a term for the purpose of tackling heterogeneity in federated networks. Our analysis (Section 3.3) is also unique in considering solving such an objective in a distributed setting with: (1) non-IID partitioned data, (2) the use of any local solver, (3) variable inexact updates across devices, and (4) a subset of devices being active at each round. These assumptions are critical to providing a characterization of such a framework in realistic federated scenarios.

In our experiments (Section 3.4), we demonstrate that tolerating partial work is beneficial in the presence of systems heterogeneity and our modified local subproblem in `FedProx` results in more robust and stable convergence compared to vanilla `FedAvg` for heterogeneous datasets. In Section 3.3, we also see that the usage of the proximal term makes `FedProx` more amenable to theoretical analysis (i.e., the local objective may be more well-behaved). In particular, if μ is chosen accordingly, the Hessian of h_k may be positive semi-definite. Hence, when F_k is non-convex, h_k will be convex, and when F_k is convex, it becomes μ-strongly convex.

Finally, we note that since `FedProx` makes only lightweight modifications to `FedAvg`, this allows us to reason about the behavior of the widely-used `FedAvg` method, and enables easy integration of `FedProx` into existing packages/systems, such as TensorFlow Federated and LEAF [1, 43]. In particular, we note that `FedAvg` is a special case of `FedProx` with (1) $\mu = 0$, (2) the local solver specifically chosen to be SGD, and (3) a constant γ (corresponding to the number of local epochs) across devices and updating rounds (i.e., no notion of systems heterogeneity). `FedProx` is in fact much more general in this regard, as it allows for partial work to be performed across devices and any local (possibly non-iterative) solver to be used on each device.

3.3 `FedProx`: Convergence Analysis

`FedAvg` and `FedProx` are stochastic algorithms by nature: in each round, only a fraction of the devices are sampled to perform the update, and the updates performed on each device may be inexact. It is well known that in order for stochastic methods to converge to a stationary point, a decreasing step-size is required. This is in contrast to non-stochastic methods, e.g., gradient descent, that can find a stationary point by employing a constant step-size. In order to analyze the convergence behavior of methods with constant step-size (as is usually implemented in practice), we need to quantify the degree of dissimilarity among the local objective functions. This could be achieved by assuming the data to be IID, i.e., homogeneous across devices. Unfortunately, in realistic federated networks, this assumption is impractical. Thus, we first propose a metric that specifically measures the dissimilarity among local functions (Section 3.3.1), and then analyze `FedProx` under this assumption while allowing for variable γ's (Section 3.3.2).

3.3.1 Local Dissimilarity

Here we introduce a measure of dissimilarity between the devices in a federated network, which is sufficient to prove convergence. This can also be satisfied via a simpler and more restrictive bounded variance assumption of the gradients, which we explore in our experiments in Section 3.4. Interestingly, similar assumptions [e.g., 250, 283, 310] have been explored elsewhere but for differing purposes; we provide a discussion of these works in Appendix 3.6.

Definition 7 (B-local dissimilarity)**.** *The local functions F_k are B-locally dissimilar at w if $\mathbb{E}_k\left[\|\nabla F_k(w)\|^2\right] \leqslant \|\nabla f(w)\|^2 B^2$. We further define $B(w) = \sqrt{\frac{\mathbb{E}_k\left[\|\nabla F_k(w)\|^2\right]}{\|\nabla f(w)\|^2}}$ for[2] $\|\nabla f(w)\| \neq 0$.*

Here $\mathbb{E}_k[\cdot]$ denotes the expectation over devices with masses $p_k = n_k/n$ and $\sum_{k=1}^N p_k = 1$ (as in Equation 3.1). Definition 7 can be seen as a generalization of the IID assumption with bounded dissimilarity, while allowing for statistical heterogeneity. As a sanity check, when all the local functions are the same, we have $B(w) = 1$ for all w. However, in the federated setting, the data distributions are often heterogeneous and $B > 1$ due to sampling discrepancies even if the samples are assumed to be IID. Let us also consider the case where $F_k(\cdot)$'s are associated with empirical risk objectives. If the samples on all the devices are homogeneous, i.e., they are sampled in an IID fashion, then as $\min_k\ n_k \to \infty$, it follows that $B(w) \to 1$ for every w as all the local functions converge to the same expected risk function in the large sample limit. Thus, $B(w) \geqslant 1$ and the larger the value of $B(w)$, the larger is the dissimilarity among the local functions.

Using Definition 7, we now state our formal dissimilarity assumption, which we use in our convergence analysis. This simply requires that the dissimilarity defined in Definition 7 is bounded. As discussed later, our convergence rate is a function of the statistical heterogeneity/device dissimilarity in the network.

Assumption 1 (Bounded dissimilarity)**.** *For some $\epsilon > 0$, there exists a B_ϵ such that for all the points $w \in \mathcal{S}_\epsilon^c = \{w \mid \|\nabla f(w)\|^2 > \epsilon\}$, $B(w) \leqslant B_\epsilon$.*

For most practical machine learning problems, there is no need to solve the problem to highly accurate stationary solutions, i.e., ϵ is typically not very small. Indeed, it is well-known that solving the problem beyond some threshold may even hurt generalization performance due to overfitting [307]. Although in practical federated learning problems the samples are not IID, they are still sampled from distributions that are not entirely unrelated (if this were the case, e.g., fitting a single global model w across devices would be ill-advised). Thus, it is reasonable to assume that the dissimilarity between local functions remains bounded throughout the training process. We also measure the dissimilarity metric empirically on real and synthetic datasets in Section 3.4.3.4 and show that this metric captures real-world statistical heterogeneity and is correlated with practical performance (the smaller the dissimilarity, the better the convergence).

[2] As an exception we define $B(w) = 1$ when $\mathbb{E}_k\left[\|\nabla F_k(w)\|^2\right] = \|\nabla f(w)\|^2$, i.e. w is a stationary solution that all the local functions F_k agree on.

3.3.2 FedProx Analysis

Using the bounded dissimilarity assumption (Assumption 1), we now analyze the amount of expected decrease in the objective when one step of FedProx is performed. Our convergence rate (Theorem 2) can be directly derived from the results of the expected decrease per updating round. We assume the same γ_k^t for any k, t for ease of notation in the following analyses.

Theorem 1 (Non-convex FedProx convergence: B-local dissimilarity)**.** *Let Assumption 1 hold. Assume the functions F_k are non-convex, L-Lipschitz smooth, and there exists $L_- > 0$, such that $\nabla^2 F_k \geq -L_- \mathbf{I}$, with $\bar{\mu} := \mu - L_- > 0$. Suppose that w^t is not a stationary solution and the local functions F_k are B-dissimilar, i.e. $B(w^t) \leqslant B$. If μ, K, and γ in Algorithm 2 are chosen such that*

$$\rho = \left(\frac{1}{\mu} - \frac{\gamma B}{\mu} - \frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} - \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right) > 0,$$

then at iteration t of Algorithm 2, we have the following expected decrease in the global objective:

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leqslant f(w^t) - \rho \|\nabla f(w^t)\|^2,$$

where S_t is the set of K devices chosen at iteration t.

We direct the reader to Appendix 3.5.1 for a detailed proof. The key steps include applying our notion of γ-inexactness (Definition 5) for each subproblem and using the bounded dissimilarity assumption, while allowing for only K devices to be active at each round. This last step in particular introduces $\mathbb{E}_{S_t}$, an expectation with respect to the choice of devices, S_t, in round t. We note that in our theory, we require $\bar{\mu} > 0$, which is a sufficient but not necessary condition for FedProx to converge. Hence, it is possible that some other μ (not necessarily satisfying $\bar{\mu} > 0$) can also enable convergence, as we explore empirically (Section 3.4).

Theorem 1 uses the dissimilarity in Definition 7 to identify sufficient decrease of the objective value at each iteration for FedProx. In Appendix 3.5.2, we provide a corollary characterizing the performance with a more common (though slightly more restrictive) bounded variance assumption. This assumption is commonly employed, e.g., when analyzing methods such as SGD. We next provide sufficient (but not necessary) conditions that ensure $\rho > 0$ in Theorem 1 such that sufficient decrease is attainable after each round.

Remark 1. *For ρ in Theorem 1 to be positive, we need $\gamma B < 1$ and $\frac{B}{\sqrt{K}} < 1$. These conditions help to quantify the tradeoff between dissimilarity (B) and the algorithm parameters (γ, K).*

Finally, we can use the above sufficient decrease to the characterize the rate of convergence to the set of approximate stationary solutions $\mathcal{S}_s = \{w \mid \mathbb{E}\left[\|\nabla f(w)\|^2\right] \leqslant \epsilon\}$ under the bounded dissimilarity assumption, Assumption 1. Note that these results hold for general non-convex $F_k(\cdot)$.

Theorem 2 (Convergence rate: FedProx)**.** *Given some $\epsilon > 0$, assume that for $B \geqslant B_\epsilon$, μ, γ, and K the assumptions of Theorem 1 hold at each iteration of FedProx. Moreover, $f(w^0) - f^* = \Delta$. Then, after $T = O(\frac{\Delta}{\rho\epsilon})$ iterations of FedProx, we have $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(w^t)\|^2\right] \leqslant \epsilon$.*

While the results thus far hold for non-convex $F_k(\cdot)$, we can also characterize the convergence for the special case of convex loss functions with exact minimization in terms of local objectives (Corollary 1). A proof is provided in Appendix 3.5.3.

Corollary 1 (Convergence: Convex case)**.** *Let the assertions of Theorem 1 hold. In addition, let $F_k(\cdot)$'s be convex and $\gamma_k^t = 0$ for any k, t, i.e., all the local problems are solved exactly, if $1 \ll B \leqslant 0.5\sqrt{K}$, then we can choose $\mu \approx 6LB^2$ from which it follows that $\rho \approx \frac{1}{24LB^2}$.*

Note that small ϵ in Assumption 1 translates to larger B_ϵ. In order to solve the problem with increasingly higher accuracies using FedProx, one needs to increase μ appropriately. We empirically verify that $\mu > 0$ leads to more stable convergence in Section 3.4.3. Moreover, in Corollary 1, if we plug in the upper bound for B_ϵ, under a bounded variance assumption, the number of required steps to achieve accuracy ϵ is $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$. Our analysis helps to characterize the performance of FedProx and similar methods when local functions are dissimilar.

Remark 2 (Comparison with SGD)**.** *We note that FedProx achieves the same asymptotic convergence guarantee as SGD: Under the bounded variance assumption, for small ϵ, if we replace B_ϵ with its upper-bound in Corollary 3 and choose μ large enough, the iteration complexity of FedProx when the subproblems are solved exactly and $F_k(\cdot)$'s are convex is $O(\frac{L\Delta}{\epsilon} + \frac{L\Delta\sigma^2}{\epsilon^2})$, the same as SGD [98].*

To provide context for the rate in Theorem 2, we compare it with SGD in the convex case in Remark 2. In general, our analysis of FedProx does not yield convergence rates that improve upon classical distributed SGD (without local updating)—even though FedProx possibly performs more work locally at each communication round. In fact, when data are generated in a non-identically distributed fashion, it is possible for local updating schemes such as FedProx to perform worse than distributed SGD. Therefore, our theoretical results do not necessarily demonstrate the superiority of FedProx over distributed SGD; rather, they provide sufficient (but not necessary) conditions for FedProx to converge. Our analysis is the first we are aware of to analyze any federated (i.e., with local-updating schemes and low device participation) optimization method for Problem (3.1) in heterogeneous settings.

Finally, we note that the previous analyses assume no systems heterogeneity and use the same γ for all devices and iterations. However, we can extend them to allow for γ to vary by device and by iteration (as in Definition 6), which corresponds to allowing devices to perform variable amounts of work as determined by the local systems conditions. We provide convergence results with variable γ's below.

Corollary 2 (Convergence: Variable γ's)**.** *Assume the functions F_k are non-convex, L-Lipschitz*

smooth, and there exists $L_- > 0$, such that $\nabla^2 F_k \geq -L_- \mathbf{I}$, with $\bar{\mu} := \mu - L_- > 0$. Suppose that w^t is not a stationary solution and the local functions F_k are B-dissimilar, i.e. $B(w^t) \leqslant B$. If μ, K, and γ_k^t in Algorithm 2 are chosen such that

$$\rho^t = \left(\frac{1}{\mu} - \frac{\gamma^t B}{\mu} - \frac{B(1+\gamma^t)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma^t)}{\bar{\mu}\mu} - \frac{L(1+\gamma^t)^2 B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma^t)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right) > 0,$$

then at iteration t of Algorithm 2, we have the following expected decrease in the global objective:

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leqslant f(w^t) - \rho^t \|\nabla f(w^t)\|^2,$$

where S_t is the set of K devices chosen at iteration t and $\gamma_t = \max_{k \in S_t} \gamma_k^t$.

The proof can be easily extended from the proof for Theorem 1 , noting the fact that $\mathbb{E}_k[(1+\gamma_k^t)\|\nabla F_k(w^t)\|] \leqslant (1+\max_{k\in S_t} \gamma_k^t)\mathbb{E}_k[\|\nabla F_k(w^t)\|]$.

3.4 Experiments

We now present empirical results for the generalized `FedProx` framework. In Section 3.4.2, we demonstrate the improved performance of `FedProx` tolerating partial solutions in the face of systems heterogeneity. In Section 3.4.3, we show the effectiveness of `FedProx` in the settings with statistical heterogeneity (regardless of systems heterogeneity). We also study the effects of statistical heterogeneity on convergence (Section 3.4.3.1) and show how empirical convergence is related to our theoretical bounded dissimilarity assumption (Assumption 1) (Section 3.4.3.4). We provide thorough details of the experimental setup in Section 3.4.1 and Appendix 3.7. All code, data, and experiments are publicly available at: github.com/litian96/FedProx.

3.4.1 Experimental Details

We evaluate `FedProx` on diverse tasks, models, and real-world federated datasets. In order to better characterize statistical heterogeneity and study its effect on convergence, we also evaluate on a set of synthetic data, which allows for more precise manipulation of statistical heterogeneity. We simulate systems heterogeneity by assigning different amounts of local work to different devices.

Synthetic Data. To generate synthetic data, we follow a similar setup to that in Shamir et al. [253], additionally imposing heterogeneity among devices. In particular, for each device k, we generate samples (X_k, Y_k) according to the model $y = argmax(\text{softmax}(Wx + b))$, $x \in \mathbb{R}^{60}, W \in \mathbb{R}^{10\times 60}, b \in \mathbb{R}^{10}$. We model $W_k \sim \mathcal{N}(u_k, 1)$, $b_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, \alpha)$; $x_k \sim \mathcal{N}(v_k, \Sigma)$, where the covariance matrix Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. Each element in the mean vector v_k is drawn from $\mathcal{N}(B_k, 1), B_k \sim N(0, \beta)$. Therefore, α controls how much local models differ from each other and β controls how much the local data at each

device differs from that of other devices. We vary α, β to generate three heterogeneous distributed datasets, denoted Synthetic (α, β), as shown in Figure 3.2. We also generate one IID dataset by setting the same W, b on all devices and setting X_k to follow the same distribution. Our goal is to learn a global W and b. Full details are given in Appendix 3.7.1.

Real Data. We also explore four real datasets; statistics are summarized in Table 3.1. These datasets are curated from prior work in federated learning as well as recent federated learning benchmarks [43, 208]. We study a convex classification problem with MNIST [165] using multinomial logistic regression. To impose statistical heterogeneity, we distribute the data among 1,000 devices such that each device has samples of only two digits and the number of samples per device follows a power law. We then study a more complex 62-class Federated Extended MNIST [43, 54] (FEMNIST) dataset using the same model. For the non-convex setting, we consider a text sentiment analysis task on tweets from Sentiment140 (Go et al., 2009) (Sent140) with an LSTM classifier, where each twitter account corresponds to a device. We also investigate the task of next-character prediction on the dataset of *The Complete Works of William Shakespeare* [208] (Shakespeare). Each speaking role in the plays is associated with a different device. Details of datasets, models, and workloads are provided in Appendix 3.7.1.

Table 3.1: Statistics of four real federated datasets.

Dataset	**Devices**	**Samples**	**Samples/device**	
			mean	stdev
MNIST	1,000	69,035	69	106
FEMNIST	200	18,345	92	159
Shakespeare	143	517,106	3,616	6,808
Sent140	772	40,783	53	32

Implementation. We implement `FedAvg` (Algorithm 1) and `FedProx` (Algorithm 2) in Tensorflow [3]. In order to draw a fair comparison with `FedAvg`, we employ SGD as a local solver for `FedProx`, and adopt a slightly different device sampling scheme than that in Algorithms 1 and 2: sampling devices uniformly and then averaging the updates with weights proportional to the number of local data points (as originally proposed in McMahan et al. [208]). While this sampling scheme is not supported by our analysis, we observe similar relative behavior of `FedProx` vs. `FedAvg` whether or not it is employed. Interestingly, we also observe that the sampling scheme proposed herein in fact results in more stable performance for both methods (see Appendix 3.7.3.4, Figure 3.12). This suggests an additional benefit of the proposed framework. Full details are provided in Appendix 3.7.2.

Hyperparameters & Evaluation Metrics. For each dataset, we tune the learning rate on FedAvg (with E=1 and without systems heterogeneity) and use the same learning rate for all experiments on that dataset. We set the number of selected devices to be 10 for all experiments on all datasets.

For each comparison, we fix the randomly selected devices, the stragglers, and mini-batch orders across all runs. We report all metrics based on the global objective $f(w)$. Note that in our simulations (see Section 3.4.2 for details), we assume that each communication round corresponds to a specific aggregation time stamp (measured in real-world global wall-clock time)—we therefore report results in terms of rounds rather than FLOPs or wall-clock time.

3.4.2 Systems Heterogeneity: Tolerating Partial Work

In order to measure the effect of allowing for partial solutions to be sent to handle systems heterogeneity with FedProx, we simulate federated settings with varying system heterogeneity, as described below.

Systems Heterogeneity Simulations. We assume that there exists a global clock during training, and each participating device determines the amount of local work as a function of this clock cycle and its systems constraints. This specified amount of local computation corresponds to some implicit value γ_k^t for device k at the t-th iteration. In our simulations, we fix a global number of epochs E, and force some devices to perform fewer updates than E epochs given their current systems constraints. In particular, for varying heterogeneous settings, at each round, we assign x number of epochs (chosen uniformly at random between [1, E]) to 0%, 50%, and 90% of the selected devices, respectively. Settings where 0% devices perform fewer than E epochs of work correspond to the environments *without* systems heterogeneity, while 90% of the devices sending their partial solutions corresponds to highly heterogeneous environments. FedAvg will simply drop these 0%, 50%, and 90% stragglers upon reaching the global clock cycle, and FedProx will incorporate the partial updates from these devices.

In Figure 3.1, we set E to be 20 and study the effects of aggregating partial work from the otherwise dropped devices. The synthetic dataset here is taken from Synthetic (1,1) in Figure 3.2. We see that on all the datasets, systems heterogeneity has negative effects on convergence, and larger heterogeneity results in worse convergence (FedAvg). Compared with dropping the more constrained devices (FedAvg), incorporating variable amounts of work (FedProx, $\mu = 0$) is beneficial and leads to more stable and faster convergence. We also observe that setting $\mu > 0$ in FedProx can further improve convergence, as we discuss in Section 3.4.3.

We additionally investigate two less heterogeneous settings. First, we limit the capability of all the devices by setting E to be 1 (i.e., all the devices run at most one local epoch), and impose systems heterogeneity in a similar way. Even in these settings, allowing for partial work can improve convergence compared with FedAvg. Second, we explore a setting without any statistical heterogeneity using an identically distributed

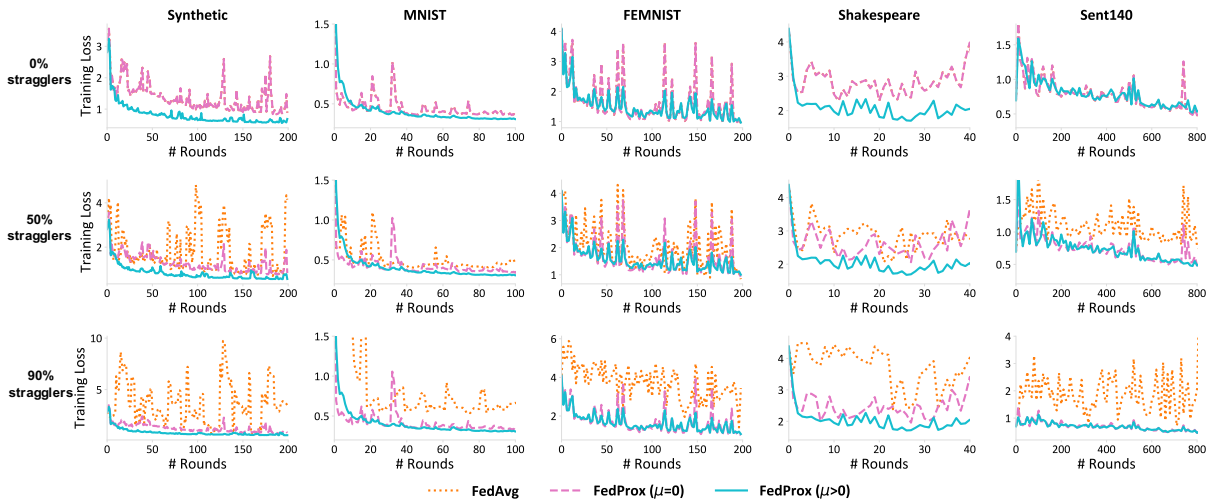


Figure 3.1: `FedProx` results in significant convergence improvements relative to `FedAvg` in heterogeneous networks. We simulate different levels of systems heterogeneity by forcing 0%, 50%, and 90% devices to be the stragglers (dropped by `FedAvg`). (1) Comparing `FedAvg` and `FedProx` ($\mu = 0$), we see that allowing for variable amounts of work to be performed can help convergence in the presence of systems heterogeneity. (2) Comparing `FedProx` ($\mu = 0$) with `FedProx` ($\mu > 0$), we show the benefits of our added proximal term. `FedProx` with $\mu > 0$ leads to more stable convergence and enables otherwise divergent methods to converge, both in the presence of systems heterogeneity (50% and 90% stragglers) and without systems heterogeneity (0% stragglers). Note that `FedProx` with $\mu = 0$ and without systems heterogeneity (no stragglers) corresponds to `FedAvg`. We also report testing accuracy in Figure 3.7, Appendix 3.7.3.2, and show that `FedProx` improves the test accuracy on all datasets.

synthetic dataset (Synthetic IID). In this IID setting, as shown in Figure 3.5 in Appendix 3.7.3.2, `FedAvg` is rather robust under device failure, and tolerating variable amounts of local work may not cause major improvement. This serves as an additional motivation to rigorously study the effect of statistical heterogeneity on new methods designed for federated learning, as simply relying on IID data (a setting unlikely to occur in practice) may not tell a complete story.

3.4.3 Statistical Heterogeneity: Proximal Term

To better understand how the proximal term can be beneficial in heterogeneous settings, we first show convergence can become worse as statistical heterogeneity increases.

3.4.3.1 Effects of Statistical Heterogeneity

In Figure 3.2 (the first row), we study how statistical heterogeneity affects convergence using four synthetic datasets without the presence of systems heterogeneity (fixing E to be 20). From left to right, as data become more heterogeneous, convergence becomes

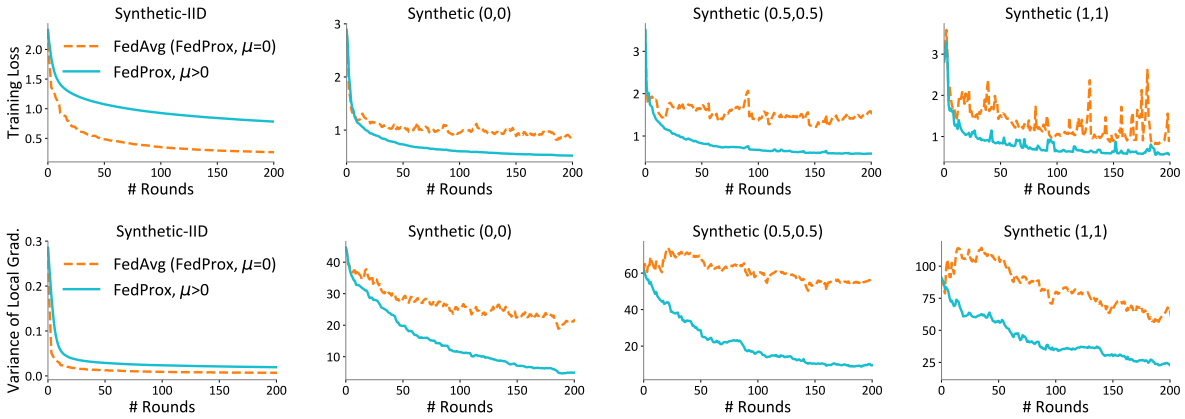


Figure 3.2: Effect of data heterogeneity on convergence. We remove the effects of systems heterogeneity by forcing each device to run the same amount of epochs. In this setting, `FedProx` with $\mu = 0$ reduces to `FedAvg`. (1) Top row: We show training loss (see results on testing accuracy in Appendix 3.7.3, Figure 3.6) on four synthetic datasets whose statistical heterogeneity increases from left to right. Note that the method with $\mu = 0$ corresponds to `FedAvg`. Increasing heterogeneity leads to worse convergence, but setting $\mu > 0$ can help to combat this. (2) Bottom row: We show the corresponding dissimilarity measurement (variance of gradients) of the four synthetic datasets. This metric captures statistical heterogeneity and is consistent with training loss—smaller dissimilarity indicates better convergence.

worse for `FedProx` with $\mu = 0$ (i.e., `FedAvg`). Though it may slow convergence for IID data, we see that setting $\mu > 0$ is particularly useful in heterogeneous settings. This indicates that the modified subproblem introduced in `FedProx` can benefit practical federated settings with varying statistical heterogeneity. For perfectly IID data, some heuristics such as decreasing μ if the loss continues to decrease may help avoid the deceleration of convergence (see Figure 3.11 in Appendix 3.7.3.3). In the sections to follow, we see similar results in our non-synthetic experiments.

3.4.3.2 Effects of $\mu > 0$

The key parameters of `FedProx` that affect performance are the amount of local work (as parameterized by the number of local epochs, E), and the proximal term scaled by μ. Intuitively, large E may cause local models to drift too far away from the initial starting point, thus leading to potential divergence [208]. Therefore, to handle the divergence or instability of `FedAvg` with non-IID data, it is helpful to tune E carefully. However, E is constrained by the underlying system's environments on the devices, and it is difficult to determine an appropriate uniform E for all devices. Alternatively, it is beneficial to allow for device-specific E's (variable γ's) and tune a best μ (a parameter that can be viewed as a re-parameterization of E) to prevent divergence and improve the stability of methods. A proper μ can restrict the trajectory of the iterates by constraining the iterates to be closer to that of the global model, thus incorporating variable amounts of updates

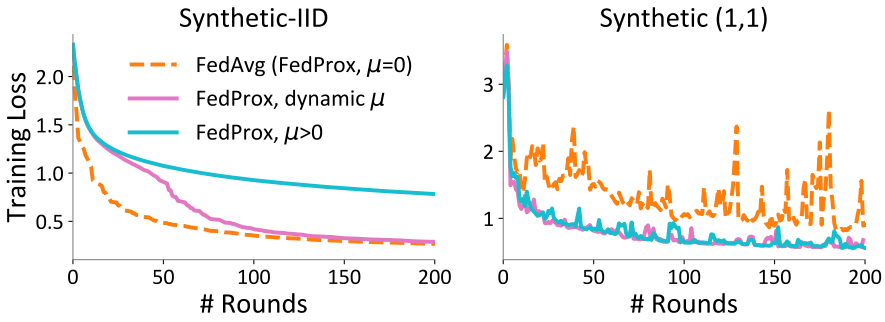


Figure 3.3: Effectiveness of setting μ adaptively based on the current model performance. We increase μ by 0.1 whenever the loss increases and decreases it by 0.1 whenever the loss decreases for 5 consecutive rounds. We initialize μ to 1 for Synthetic IID (in order to be adversarial to our methods), and initialize μ to 0 for Synthetic (1,1). This simple heuristic works well empirically.

and guaranteeing convergence (Theorem 2).

We show the effects of the proximal term in `FedProx` ($\mu > 0$) in Figure 3.1. For each experiment, we compare the results between `FedProx` with $\mu = 0$ and `FedProx` with a best μ (see the next paragraph for discussions on how to select μ). For all datasets, we observe that the appropriate μ can increase the stability for unstable methods and can force divergent methods to converge. This holds both when there is systems heterogeneity (50% and 90% stragglers) and there is no systems heterogeneity (0% stragglers). $\mu > 0$ also increases the accuracy in most cases (see Figure 3.6 and Figure 3.7 in Appendix 3.7.3.2). In particular, `FedProx` improves absolute testing accuracy relative to `FedAvg` by 22% on average in highly heterogeneous environments (90% stragglers).

3.4.3.3 Choosing μ.

One natural question is to determine how to set the penalty constant μ in the proximal term. A large μ may potentially slow the convergence by forcing the updates to be close to the starting point, while a small μ may not make any difference. In all experiments, we tune the best μ from the limited candidate set $\{0.001, 0.01, 0.1, 1\}$. For the five federated datasets in Figure 3.1, the best μ values are 1, 1, 1, 0.001, and 0.01, respectively. While automatically tuning μ is difficult to instantiate directly from our theoretical results, in practice, we note that μ can be adaptively chosen based on the current performance of the model. For example, one simple heuristic is to increase μ when seeing the loss increasing and decreasing μ when seeing the loss decreasing. In Figure 3.3, we demonstrate the effectiveness of this heuristic using two synthetic datasets. Note that we start from initial μ values that are adversarial to our methods. We provide full results showing the competitive performance of this approach in Appendix 3.7.3.3. Future work includes developing methods to automatically tune this parameter for heterogeneous datasets, based, e.g., on the theoretical groundwork provided here.

3.4.3.4 Dissimilarity Measurement and Divergence

Finally, in Figure 3.2 (the bottom row), we demonstrate that our B-local dissimilarity measurement in Definition 7 captures the heterogeneity of datasets and is therefore an appropriate proxy of performance. In particular, we track the variance of gradients on each device, $E_k[\|\nabla F_k(w) - \nabla f(w)\|^2]$, which is lower bounded by B_ϵ. Empirically, we observe that increasing μ leads to smaller dissimilarity among local functions F_k, and that the dissimilarity metric is consistent with the training loss. Therefore, smaller dissimilarity indicates better convergence, which can be enforced by setting μ appropriately. We also show the dissimilarity metric on real federated data in Appendix 3.7.3.2.

3.5 Complete Proofs

3.5.1 Proof of Theorem 1

Proof. Using our notion of γ-inexactness for each local solver (Definition 5), we can define e_k^{t+1} such that:

$$\begin{aligned} &\nabla F_k(w_k^{t+1}) + \mu(w_k^{t+1} - w^t) - e_k^{t+1} = 0, \\ &\|e_k^{t+1}\| \leqslant \gamma \|\nabla F_k(w^t)\|. \end{aligned} \tag{3.3}$$

Now let us define $\bar{w}^{t+1} = \mathbb{E}_k\Big[w_k^{t+1}\Big]$. Based on this definition, we know

$$\bar{w}^{t+1} - w^t = \frac{-1}{\mu}\mathbb{E}_k\Big[\nabla F_k(w_k^{t+1})\Big] + \frac{1}{\mu}\mathbb{E}_k\Big[e_k^{t+1}\Big]. \tag{3.4}$$

Let us define $\bar{\mu} = \mu - L_- > 0$ and $\hat{w}_k^{t+1} = \arg\min_w \; h_k(w; w^t)$. Then, due to the $\bar{\mu}$-strong convexity of h_k, we have

$$\|\hat{w}_k^{t+1} - w_k^{t+1}\| \leqslant \frac{\gamma}{\bar{\mu}}\|\nabla F_k(w^t)\|. \tag{3.5}$$

Note that once again, due to the $\bar{\mu}$-strong convexity of h_k, we know that $\|\hat{w}_k^{t+1} - w^t\| \leqslant \frac{1}{\bar{\mu}}\|\nabla F_k(w^t)\|$. Now we can use the triangle inequality to get

$$\|w_k^{t+1} - w^t\| \leqslant \frac{1+\gamma}{\bar{\mu}}\|\nabla F_k(w^t)\|. \tag{3.6}$$

Therefore,

$$\begin{aligned} \|\bar{w}^{t+1} - w^t\| &\leqslant \mathbb{E}_k\Big[\|w_k^{t+1} - w^t\|\Big] \leqslant \frac{1+\gamma}{\bar{\mu}}\mathbb{E}_k\big[\|\nabla F_k(w^t)\|\big] \\ &\leqslant \frac{1+\gamma}{\bar{\mu}}\sqrt{\mathbb{E}_k[\|\nabla F_k(w^t)\|^2]} \leqslant \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|, \end{aligned} \tag{3.7}$$

where the last inequality is due to the bounded dissimilarity assumption.
Now let us define M_{t+1} such that $\bar{w}^{t+1} - w^t = \frac{-1}{\mu}\big(\nabla f(w^t) + M_{t+1}\big)$, i.e.,

$$M_{t+1} = \mathbb{E}_k\Big[\nabla F_k(w_k^{t+1}) - \nabla F_k(w^t) - e_k^{t+1}\Big]. \tag{3.8}$$

We can bound $\|M_{t+1}\|$:

$$\begin{aligned} \|M_{t+1}\| &\leqslant \mathbb{E}_k\Big[L\|w_k^{t+1} - w_k^t\| + \|e_k^{t+1}\|\Big] \\ &\leqslant \left(\frac{L(1+\gamma)}{\bar{\mu}} + \gamma\right) \times \mathbb{E}_k\big[\|\nabla F_k(w^t)\|\big] \leqslant \left(\frac{L(1+\gamma)}{\bar{\mu}} + \gamma\right) B\|\nabla f(w^t)\|, \end{aligned} \tag{3.9}$$

where the last inequality is also due to bounded dissimilarity assumption. Based on the L-Lipschitz smoothness of f and Taylor expansion, we have

$$\begin{aligned} f(\bar{w}^{t+1}) &\leqslant f(w^t) + \langle \nabla f(w^t), \bar{w}^{t+1} - w^t \rangle + \frac{L}{2}\|\bar{w}^{t+1} - w^t\|^2 \\ &\leqslant f(w^t) - \frac{1}{\mu}\|\nabla f(w^t)\|^2 - \frac{1}{\mu}\langle \nabla f(w^t), M_{t+1} \rangle + \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2}\|\nabla f(w^t)\|^2 \\ &\leqslant f(w^t) - \left(\frac{1-\gamma B}{\mu} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} - \frac{L(1+\gamma)^2 B^2}{2\bar{\mu}^2} \right) \times \|\nabla f(w^t)\|^2. \end{aligned} \tag{3.10}$$

From the above inequality it follows that if we set the penalty parameter μ large enough, we can get a decrease in the objective value of $f(\bar{w}^{t+1}) - f(w^t)$ which is proportional to $\|\nabla f(w^t)\|^2$. However, this is not the way that the algorithm works. In the algorithm, we only use K devices that are chosen randomly to approximate $\bar{w}^t$. So, in order to find the $\mathbb{E}\big[f(w^{t+1})\big]$, we use local Lipschitz continuity of the function f.

$$f(w^{t+1}) \leqslant f(\bar{w}^{t+1}) + L_0\|w^{t+1} - \bar{w}^{t+1}\|, \tag{3.11}$$

where L_0 is the local Lipschitz continuity constant of function f and we have

$$L_0 \leqslant \|\nabla f(w^t)\| + L\max(\|\bar{w}^{t+1} - w^t\|, \|w^{t+1} - w^t\|) \leqslant \|\nabla f(w^t)\| + L(\|\bar{w}^{t+1} - w^t\| + \|w^{t+1} - w^t\|).$$

Therefore, if we take expectation with respect to the choice of devices in round t we need to bound

$$\mathbb{E}_{S_t}\Big[f(w^{t+1})\Big] \leqslant f(\bar{w}^{t+1}) + Q_t, \tag{3.12}$$

where $Q_t = \mathbb{E}_{S_t}\big[L_0\|w^{t+1} - \bar{w}^{t+1}\|\big]$. Note that the expectation is taken over the random choice of devices to update.

$$\begin{aligned} Q_t &\leqslant \mathbb{E}_{S_t}\left[\Big(\|\nabla f(w^t)\| + L(\|\bar{w}^{t+1} - w^t\| + \|w^{t+1} - w^t\|) \Big) \times \|w^{t+1} - \bar{w}^{t+1}\| \right] \\ &\leqslant \Big(\|\nabla f(w^t)\| + L\|\bar{w}^{t+1} - w^t\| \Big) \mathbb{E}_{S_t}\Big[\|w^{t+1} - \bar{w}^{t+1}\|\Big] + L\mathbb{E}_{S_t}\Big[\|w^{t+1} - w^t\| \cdot \|w^{t+1} - \bar{w}^{t+1}\|\Big] \\ &\leqslant \Big(\|\nabla f(w^t)\| + 2L\|\bar{w}^{t+1} - w^t\| \Big) \mathbb{E}_{S_t}\Big[\|w^{t+1} - \bar{w}^{t+1}\|\Big] + L\mathbb{E}_{S_t}\Big[\|w^{t+1} - \bar{w}^{t+1}\|^2\Big] \end{aligned} \tag{3.13}$$

From (3.7), we have that $\|\bar{w}^{t+1} - w^t\| \leqslant \frac{B(1+\gamma)}{\bar{\mu}}\|\nabla f(w^t)\|$. Moreover,

$$\mathbb{E}_{S_t}\Big[\|w^{t+1} - \bar{w}^{t+1}\|\Big] \leqslant \sqrt{\mathbb{E}_{S_t}\big[\|w^{t+1} - \bar{w}^{t+1}\|^2\big]} \tag{3.14}$$

and

$$\mathbb{E}_{S_t}\Big[\|w^{t+1} - \bar{w}^{t+1}\|^2\Big] \leqslant \frac{1}{K}\mathbb{E}_k\Big[\|w_k^{t+1} - \bar{w}^{t+1}\|^2\Big]$$

$$
\begin{aligned}
&\leqslant \frac{2}{K}\mathbb{E}_k\Big[\|w_k^{t+1} - w^t\|^2\Big], \quad (\text{as } \bar{w}^{t+1} = \mathbb{E}_k\Big[w_k^{t+1}\Big]) \\
&\leqslant \frac{2}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\mathbb{E}_k\Big[\|\nabla F_k(w^t)\|^2\Big] \quad (\text{from (3.6)}) \\
&\leqslant \frac{2B^2}{K}\frac{(1+\gamma)^2}{\bar{\mu}^2}\|\nabla f(w^t)\|^2, \qquad (3.15)
\end{aligned}
$$

where the first inequality is a result of K devices being chosen randomly to get w^t and the last inequality is due to bounded dissimilarity assumption. If we replace these bounds in (3.13) we get

$$
Q_t \leqslant \left(\frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} + \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\right)\|\nabla f(w^t)\|^2 \qquad (3.16)
$$

Combining (3.10), (3.12), (3.11) and (3.16) and using the notation $\alpha = \frac{1}{\mu}$ we get

$$
\begin{aligned}
\mathbb{E}_{S_t}\Big[f(w^{t+1})\Big] \leqslant f(w^t) - \Bigg(&\frac{1}{\mu} - \frac{\gamma B}{\mu} - \frac{B(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} - \frac{LB(1+\gamma)}{\bar{\mu}\mu} \\
&- \frac{L(1+\gamma)^2B^2}{2\bar{\mu}^2} - \frac{LB^2(1+\gamma)^2}{\bar{\mu}^2 K}\left(2\sqrt{2K}+2\right)\Bigg)\|\nabla f(w^t)\|^2.
\end{aligned}
$$

□

3.5.2 Proof for Bounded Variance

Corollary 3 (Bounded variance equivalence)**.** *Let Assumption 1 hold. Then, in the case of bounded variance, i.e.,* $\mathbb{E}_k\big[\|\nabla F_k(w) - \nabla f(w)\|^2\big] \leqslant \sigma^2$*, for any* $\epsilon > 0$ *it follows that* $B_\epsilon \leqslant \sqrt{1 + \frac{\sigma^2}{\epsilon}}$.

Proof. We have,

$$
\begin{aligned}
&E_k[\|\nabla F_k(w) - \nabla f(w)\|^2] = E_k[\|\nabla F_k(w)\|^2] - \|\nabla f(w)\|^2 \leqslant \sigma^2 \\
&\Rightarrow E_k[\|\nabla F_k(w)\|^2] \leqslant \sigma^2 + \|\nabla f(w)\|^2 \\
&\Rightarrow B_\epsilon = \sqrt{\frac{E_k[\|\nabla F_k(w)\|^2]}{\|\nabla f(w)\|^2}} \leqslant \sqrt{1 + \frac{\sigma^2}{\epsilon}}.
\end{aligned}
$$

With Corollary 3 in place, we can restate the main result in Theorem 1 in terms of the bounded variance assumption.

Theorem 3 (Non-convex `FedProx` convergence: Bounded variance)**.** *Let the assertions of Theorem 1 hold. In addition, let the iterate* w^t *be such that* $\left\|\nabla f(w^t)\right\|^2 \geqslant \epsilon$*, and let*

$\mathbb{E}_k\left[\|\nabla F_k(w) - \nabla f(w)\|^2\right] \leqslant \sigma^2$ *hold instead of the dissimilarity condition. If* μ, K *and* γ *in Algorithm 2 are chosen such that*

$$\rho = \left(\frac{1}{\mu} - \left(\frac{\gamma}{\mu} + \frac{(1+\gamma)\sqrt{2}}{\bar{\mu}\sqrt{K}} + \frac{L(1+\gamma)}{\bar{\mu}\mu}\right)\sqrt{1+\frac{\sigma^2}{\epsilon}} - \left(\frac{L(1+\gamma)^2}{2\bar{\mu}^2} + \frac{L(1+\gamma)^2}{\bar{\mu}^2 K}(2\sqrt{2K}+2)\right)\left(1+\frac{\sigma^2}{\epsilon}\right)\right) > 0,$$

then at iteration t *of Algorithm 2, we have the following expected decrease in the global objective:*

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \leqslant f(w^t) - \rho\|\nabla f(w^t)\|^2,$$

where S_t *is the set of* K *devices chosen at iteration* t*.*

The proof of Theorem 3 follows from the proof of Theorem 1 by noting the relationship between the bounded variance assumption and the dissimilarity assumption as portrayed by Corollary 3.

3.5.3 Proof of Corollary 1

In the convex case, where $L_- = 0$ and $\bar{\mu} = \mu$, if $\gamma = 0$, i.e., all subproblems are solved accurately, we can get a decrease proportional to $\|\nabla f(w^t)\|^2$ if $B < \sqrt{K}$. In such a case if we assume $1 << B \leqslant 0.5\sqrt{K}$, then we can write

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \lessapprox f(w^t) - \frac{1}{2\mu}\|\nabla f(w^t)\|^2 + \frac{3LB^2}{2\mu^2}\|\nabla f(w^t)\|^2 . \tag{3.17}$$

In this case, if we choose $\mu \approx 6LB^2$ we get

$$\mathbb{E}_{S_t}\left[f(w^{t+1})\right] \lessapprox f(w^t) - \frac{1}{24LB^2}\|\nabla f(w^t)\|^2 . \tag{3.18}$$

Note that the expectation in (3.18) is a conditional expectation conditioned on the previous iterate. Taking expectation of both sides, and telescoping, we have that the number of iterations to at least generate one solution with squared norm of gradient less than ϵ is $O(\frac{LB^2\Delta}{\epsilon})$.

3.6 Connections to Other Single-Machine and Distributed Methods

Two aspects of the proposed work—the proximal term in `FedProx`, and the bounded dissimilarity assumption used in our analysis—have been previously studied in the optimization literature, but with very different motivations. For completeness, we provide a discussion below on our relation to these prior works.

Proximal Term. The proposed modified objective in `FedProx` shares a connection with elastic averaging SGD (EASGD) [323], which was proposed as a way to train deep networks in the data center setting, and uses a similar proximal term in its objective. While the intuition is similar to EASGD (this term helps to prevent large deviations on each device/machine), EASGD employs a more complex moving average to update parameters, is limited to using SGD as a local solver, and has only been analyzed for simple quadratic problems. The proximal term we introduce has also been explored in previous optimization literature with different purposes, such as Allen-Zhu [14], to speed up (mini-batch) SGD training on a single machine, and in Li et al. [173] for efficient SGD training both in a single machine and distributed settings. However, the analysis in Li et al. [173] is limited to a single machine setting with different assumptions (e.g., IID data and solving the subproblem exactly at each round).

In addition, DANE [253] and AIDE [236], distributed methods designed for the data center setting, propose a similar proximal term in the local objective function, but also augment this with an additional gradient correction term. Both methods assume that all devices participate at each communication round, which is impractical in federated settings. Indeed, due to the inexact estimation of full gradients (i.e., $\nabla \phi(w^{(t-1)})$ in Shamir et al. [253, Eq (13)]) with device subsampling schemes and the staleness of the gradient correction term [253, Eq (13)], these methods are not directly applicable to our setting. Regardless of this, we explore a variant of such an approach in federated settings and see that the gradient direction term does not help in this scenario—performing uniformly worse than the proposed `FedProx` framework for heterogeneous datasets, despite the extra computation required (see Figure 3.4). We refer interested readers to Li et al. [180] for more detailed discussions.

Finally, we note that there is an interesting connection between meta-learning methods and federated optimization methods [153], and similar proximal terms have recently been investigated in the context of meta-learning for improved performance on few-shot learning tasks [102, 330].

Bounded Dissimilarity Assumption. The bounded dissimilarity assumption we discuss in Assumption 1 has appeared in different forms, for example in Schmidt and Roux [250], Vaswani et al. [283], Yin et al. [310]. In Yin et al. [310], the bounded similarity assumption is used in the context of asserting gradient diversity and quantifying the benefit in terms of scaling of the mean square error for mini-batch SGD for IID data. In Schmidt and Roux [250], Vaswani et al. [283], the authors use a similar assumption, called *strong growth condition*, which is a stronger version of Assumption 1 with $\epsilon = 0$. They prove that some interesting practical problems satisfy such a condition. They also use this assumption to prove optimal and better convergence rates for SGD with constant step-sizes. Note that this is different from our approach as the algorithm that we are analyzing is not SGD, and our analysis is different in spite of the similarity in the assumptions.

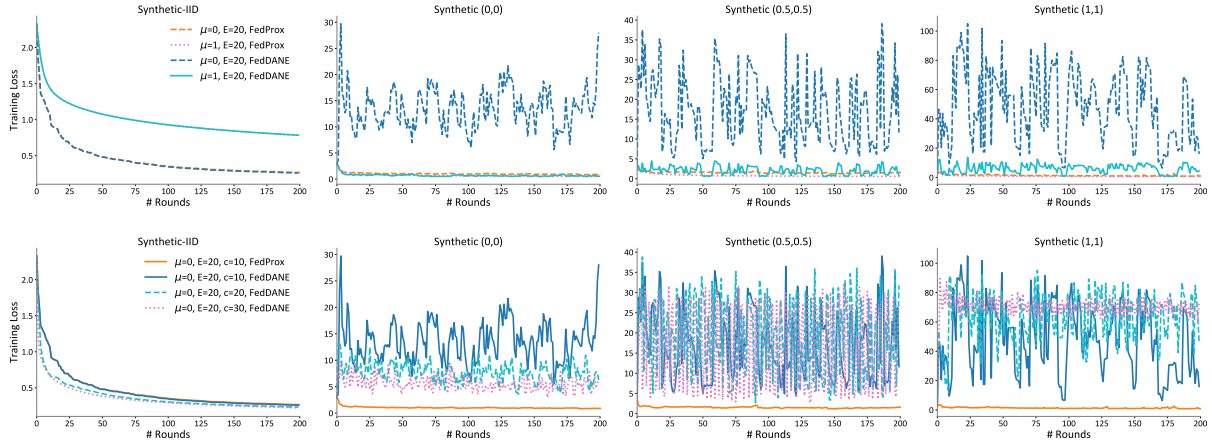


Figure 3.4: DANE and AIDE [236, 253] are methods proposed in the data center setting that use a similar proximal term as `FedProx` as well as an additional gradient correction term. We modify DANE to apply to federated settings by allowing for local updating and low participation of devices. We show the convergence of this modified method, which we call `FedDane`, on synthetic datasets. In the top figures, we subsample 10 devices out of 30 on all datasets for both `FedProx` and `FedDane`. While `FedDane` performs similarly as `FedProx` on the IID data, it suffers from poor convergence on the non-IID datasets. In the bottom figures, we show the results of `FedDane` when we increase the number of selected devices in order to narrow the gap between our estimated full gradient and the real full gradient (in the gradient correction term). Note that communicating with all (or most of the) devices is already unrealistic in practical settings. We observe that although sampling more devices per round might help to some extent, `FedDane` is still unstable and tends to diverge. This serves as additional motivation for the specific subproblem we propose in `FedProx`.

3.7 Simulation Details and Additional Experiments

3.7.1 Datasets and Models

Here we provide full details on the datasets and models used in our experiments. We curate a diverse set of non-synthetic datasets, including those used in prior work on federated learning [208], and some proposed in LEAF, a benchmark for federated settings [43]. We also create synthetic data to directly test the effect of heterogeneity on convergence, as in Section 3.4.1.

- **Synthetic:** We set (α, β)=(0,0), (0.5,0.5) and (1,1) respectively to generate three non-identical distributed datasets (Figure 3.2). In the IID data (Figure 3.5), we set the same $W, b \sim \mathcal{N}(0,1)$ on all devices and X_k to follow the same distribution $\mathcal{N}(v, \Sigma)$ where each element in the mean vector v is zero and Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. For all synthetic datasets, there are 30 devices in total and the number of samples on each device follows a power law.

- **MNIST:** We study image classification of handwritten digits 0-9 in MNIST [165] using multinomial logistic regression. To simulate a heterogeneous setting, we distribute the data among 1000 devices such that each device has samples of only 2 digits and the number of samples per device follows a power law. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.
- **FEMNIST:** We study an image classification problem on the 62-class EMNIST dataset [54] using multinomial logistic regression. To generate heterogeneous data partitions, we subsample 10 lower case characters ('a'-'j') from EMNIST and distribute only 5 classes to each device. We call this *federated* version of EMNIST *FEMNIST*. There are 200 devices in total. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.
- **Shakespeare:** This is a dataset built from *The Complete Works of William Shakespeare* [208]. Each speaking role in a play represents a different device. We use a two-layer LSTM classifier containing 100 hidden units with an 8D embedding layer. The task is next-character prediction, and there are 80 classes of characters in total. The model takes as input a sequence of 80 characters, embeds each of the characters into a learned 8-dimensional space and outputs one character per training sample after 2 LSTM layers and a densely-connected layer.
- **Sent140:** In non-convex settings, we consider a text sentiment analysis task on tweets from Sentiment140 [101] (Sent140) with a two layer LSTM binary classifier containing 256 hidden units with pretrained 300D GloVe embedding [228]. Each twitter account corresponds to a device. The model takes as input a sequence of 25 characters, embeds each of the characters into a 300-dimensional space by looking up Glove and outputs one character per training sample after 2 LSTM layers and a densely-connected layer.

3.7.2 Implementation Details

Implementation. In order to draw a fair comparison with `FedAvg`, we use SGD as a local solver for `FedProx`, and adopt a slightly different device sampling scheme than that in Algorithms 1 and 2: sampling devices uniformly and averaging updates with weights proportional to the number of local data points (as originally proposed in McMahan et al. [208]). While this sampling scheme is not supported by our analysis, we observe similar relative behavior of `FedProx` vs. `FedAvg` whether or not it is employed (Figure 3.12). Interestingly, we also observe that the sampling scheme proposed herein results in more stable performance for both methods. This suggests an added benefit of the proposed framework.

Machines. We simulate the federated learning setup (1 server and N devices) on a commodity machine with 2 Intel® Xeon® E5-2650 v4 CPUs and 8 NVidia® 1080Ti GPUs.

Hyperparameters. We randomly split the data on each local device into an 80% training set and a 20% testing set. We fix the number of selected devices per round to be 10 for all experiments on all datasets. We also do a grid search on the learning rate based on `FedAvg`. We do not decay the learning rate through all rounds. For all synthetic data experiments, the learning rate is 0.01. For MNIST, FEMNIST, Shakespeare, and Sent140, we use the learning rates of 0.03, 0.003, 0.8, and 0.3. We use a batch size of 10 for all experiments.

Libraries. All code is implemented in Tensorflow Version 1.10.1 [3].
Please see github.com/litian96/FedProx for full details.

3.7.3 Additional Experiments and Full Results

3.7.3.1 Effects of Systems Heterogeneity on IID Data

We show the effects of allowing for partial work on a perfect IID synthetic data (Synthetic IID).

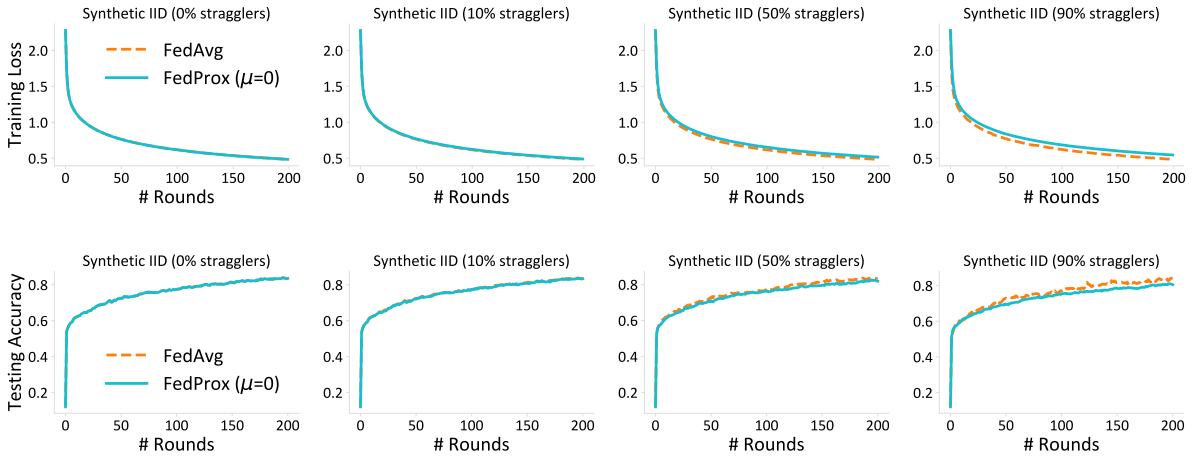


Figure 3.5: `FedAvg` is robust to device failure with IID data. In this case, whether incorporating partial solutions from the stragglers would not have much effect on convergence.

3.7.3.2 Complete Results

In Figure 3.6, we present testing accuracy on four synthetic datasets associated with the experiments shown in Figure 3.2.

In Figure 3.7, we show the testing accuracy associated with the experiments described in Figure 3.1. We calculate the accuracy improvement numbers by identifying the accuracies of `FedProx` and `FedAvg` when they have either converged, started to diverge, or run sufficient number of rounds (e.g., 1000 rounds), whichever comes earlier. We consider the methods to converge when the loss difference in two consecutive rounds

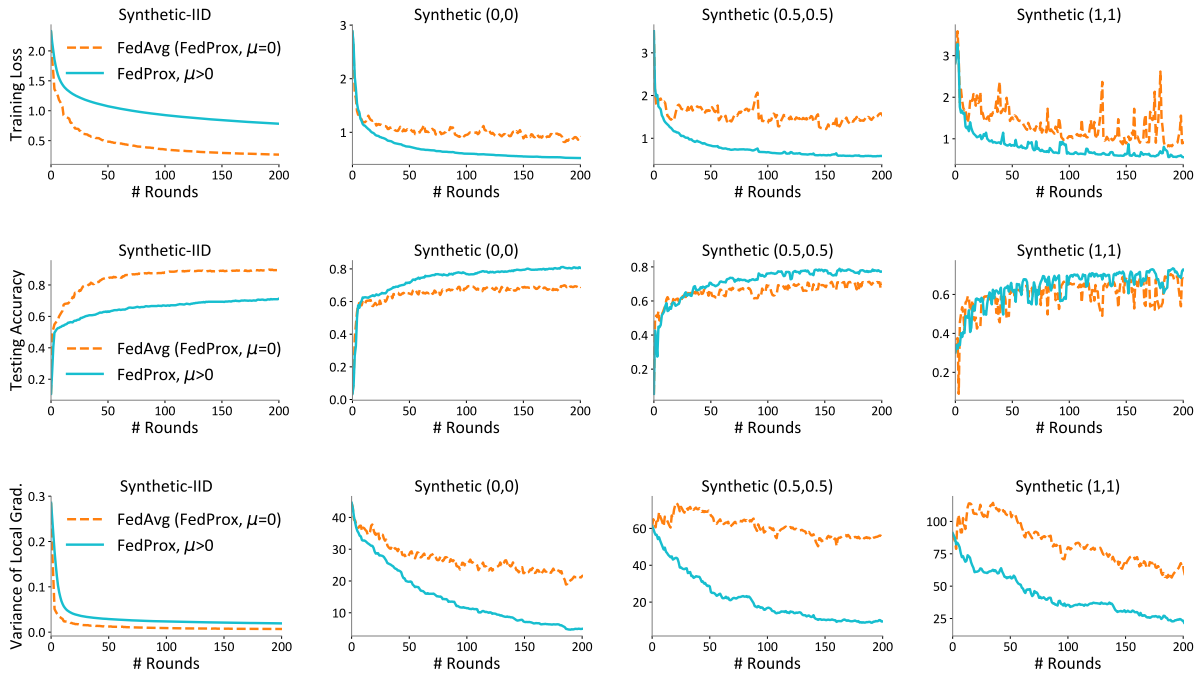


Figure 3.6: Training loss, test accuracy, and dissimilarity measurement for experiments described in Fig. 3.2.

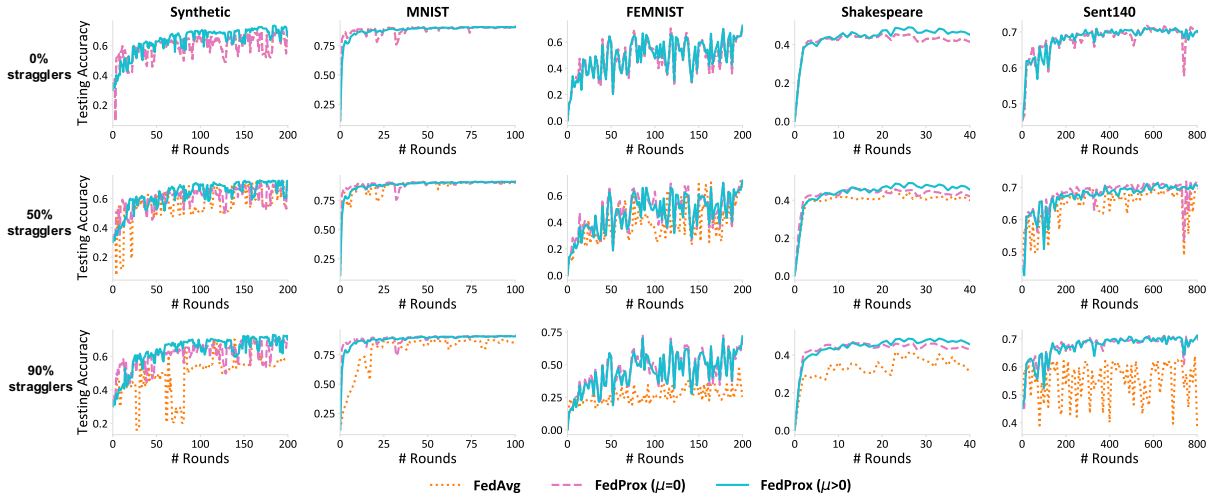


Figure 3.7: The testing accuracy of the experiments in Figure 3.1. `FedProx` achieves on average 22% improvement in terms of testing accuracy in highly heterogeneous settings (90% stragglers).

$|f_t - f_{t-1}|$ is smaller than 0.0001, and consider the methods to diverge when we see $f_t - f_{t-10}$ greater than 1.

In Figure 3.8, we report the dissimilarity measurement on five datasets (including four real datasets) described in Figure 3.1. Again, the dissimilarity characterization is

consistent with the real performance (the loss).

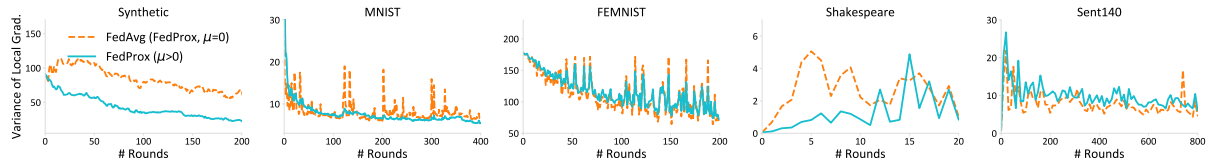


Figure 3.8: The dissimilarity metric on five datasets in Figure 3.1. We remove systems heterogeneity by only considering the case when no participating devices drop out of the network. Our dissimilarity assumption captures the data heterogeneity and is consistent with practical performance (see training loss in Figure 3.1).

In Figure 3.9 and Figure 3.10, we show the effects (both loss and testing accuracy) of allowing for partial solutions under systems heterogeneity when $E = 1$ (i.e., the statistical heterogeneity is less likely to affect convergence negatively).

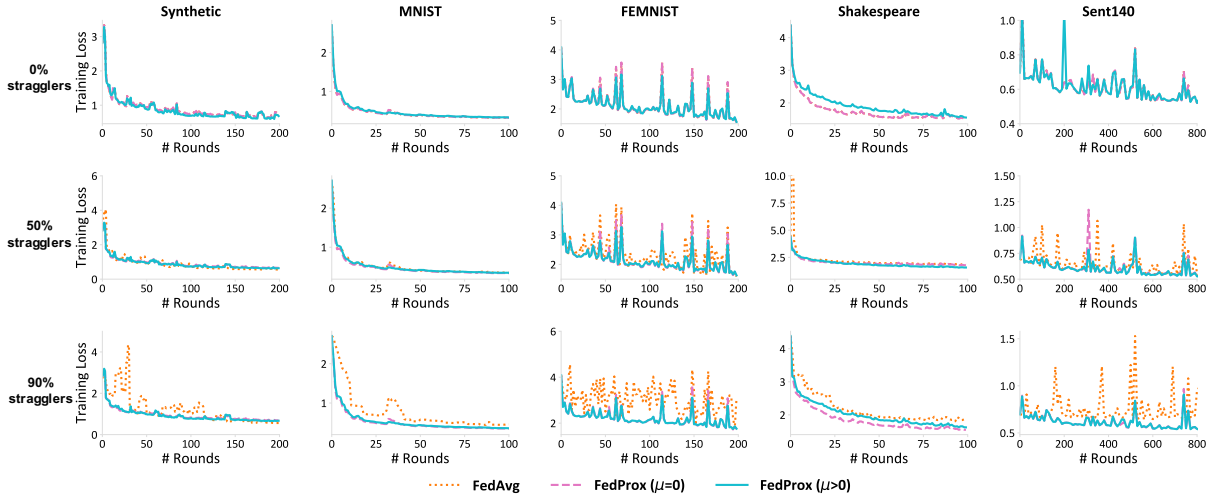


Figure 3.9: The loss of `FedAvg` and `FedProx` under various systems heterogeneity settings when each device can run at most 1 epoch at each iteration ($E = 1$). Since local updates will not deviate too much from the global model compared with the deviation under large E's, it is less likely that the statistical heterogeneity will affect convergence negatively. Tolerating for partial solutions to be sent to the central server (`FedProx`, $\mu = 0$) still performs better than dropping the stragglers (`FedAvg`).

3.7.3.3 Adaptively Setting μ

One of the key parameters of `FedProx` is μ. We provide the complete results of a simple heuristic of adaptively setting μ on four synthetic datasets in Figure 3.11. For the IID dataset (Synthetic-IID), μ starts from 1, and for the other non-IID datasets, μ starts from 0. Such initialization is adversarial to our methods. We decrease μ by 0.1 when the loss

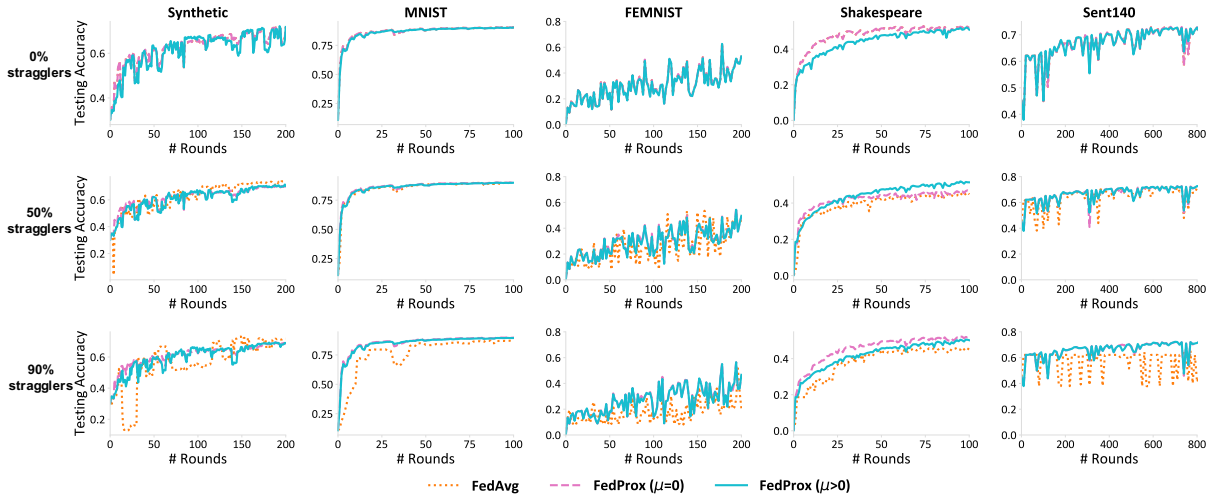


Figure 3.10: The testing accuracy of the experiments shown in Figure 3.9.

continues to decrease for 5 rounds and increase μ by 0.1 when we see the loss increase. This heuristic allows for competitive performance. It could also alleviate the potential issue that $\mu > 0$ might slow down convergence on IID data, which rarely occurs in real federated settings.

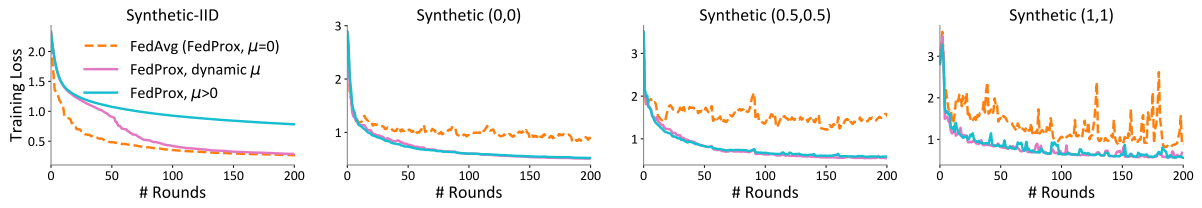


Figure 3.11: Full results of choosing μ adaptively on all the synthetic datasets. We increase μ by 0.1 whenever the loss increases and decreases it by 0.1 whenever the loss decreases for 5 consecutive rounds. We initialize μ to 1 for the IID data (Synthetic-IID) (in order to be adversarial to our methods), and initialize it to 0 for the other three non-IID datasets. We observe that this simple heuristic works well in practice.

3.7.3.4 Comparing Two Device Sampling Schemes

We show the training loss, testing accuracy, and dissimilarity measurement of `FedProx` on a set of synthetic data using two different device sampling schemes in Figure 3.12. Since our goal is to compare these two sampling schemes, we let each device perform the uniform amount of work ($E = 20$) for both methods.

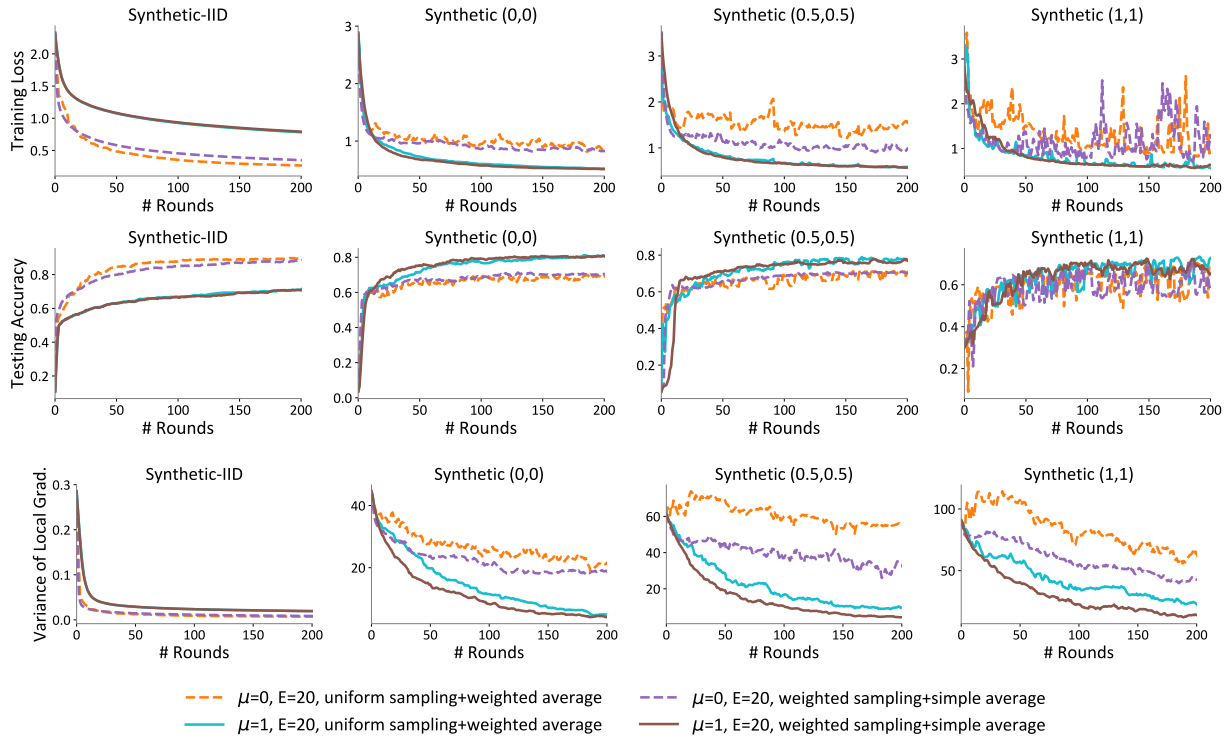


Figure 3.12: Differences between two sampling schemes in terms of training loss, testing accuracy, and dissimilarity measurement. Sampling devices with a probability proportional to the number of local data points and then simply averaging local models performs slightly better than uniformly sampling devices and averaging the local models with weights proportional to the number of local data points. Under either sampling scheme, the settings with $\mu = 1$ demonstrate more stable performance than settings with $\mu = 0$.

Chapter 4

Fairness: Addressing Representation Bias in Federated Learning

Beyond accuracy or optimization, heterogeneity also affects fairness in terms of providing fair quality of service for all participants in the network. In this chapter, we discuss a fair learning objective, along with efficient solvers. The objective offers flexible and scalable fairness/utility tradeoffs, which has broader applications where min-max objectives are desired but expensive to optimize.

4.1 Overview

In federated learning, naively minimizing an aggregate loss in a large network may disproportionately advantage or disadvantage the model performance on some of the devices. For example, although the accuracy may be high on average, there is no accuracy guarantee for individual devices in the network. This is exacerbated by the fact that the data are often heterogeneous in federated networks both in terms of size and distribution, and model performance can thus vary widely. In this work, we therefore ask: Can we devise an efficient federated optimization method to encourage a *more fair (i.e., more uniform) distribution* of the model performance across devices in federated networks?

There has been tremendous recent interest in developing fair methods for machine learning [see, e.g., 59, 83]. However, current approaches do not adequately address concerns in the federated setting. For example, a common definition in the fairness literature is to enforce *accuracy parity* between protected groups[1] [315]. For devices in massive federated networks, however, it does not make sense for the accuracy to be *identical* on each device given the significant variability of data in the network. Recent work has taken a step towards addressing this by introducing *good-intent fairness*, in which the goal is instead to ensure that the training procedure does not overfit a model to any one device at the expense of another [215]. However, the proposed objective is rigid in

[1]While fairness is typically concerned with performance between "groups", we define fairness in the federated setting at a more granular scale in terms of the devices in the network. We note that devices may naturally combine to form groups, and thus use these terms interchangeably in the context of prior work.

the sense that it only maximizes the performance of the worst performing device/group, and has only be tested in small networks (for 2-3 devices). In realistic federated learning applications, it is natural to instead seek methods that can flexibly trade off between overall performance and fairness in the network, and can be implemented at scale across hundreds to millions of devices.

In this work, we propose q-FFL, a novel optimization objective that addresses fairness issues in federated learning. Inspired by work in fair resource allocation for wireless networks, q-FFL minimizes an aggregate *reweighted* loss parameterized by q such that the devices with higher loss are given higher relative weight. We show that this objective encourages a device-level definition of fairness in the federated setting, which generalizes standard accuracy parity by measuring the degree of uniformity in performance across devices.

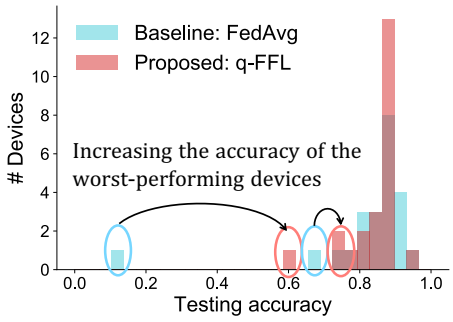


Figure 4.1: Model performance (e.g., test accuracy) in federated networks can vary widely across devices. Our objective, q-FFL, aims to increase the fairness/uniformity of model performance while maintaining average performance.

As a motivating example, we examine the test accuracy distribution of a model trained via a baseline approach (FedAvg) vs. q-FFL in Figure 4.1. Due to the variation in the data across devices, the model accuracy is quite poor on some devices. By using q-FFL, we can maintain the same overall average accuracy while ensuring a more fair/uniform quality of service across the network. Adaptively minimizing our q-FFL objective results in a flexible framework that can be tuned depending on the desired amount of fairness.

To solve q-FFL in massive federated networks, we additionally propose a lightweight and scalable distributed method, q-FedAvg. Our method carefully accounts for important characteristics of the federated setting such as communication-efficiency and low participation of devices [36, 209]. The method also reduces the overhead of tuning the hyperparameter q in q-FFL by dynamically estimating the step-sizes associated with different values of q.

Through extensive experiments on federated datasets with both convex and non-convex models, we demonstrate the fairness and flexibility of q-FFL and the efficiency of q-FedAvg compared with existing baselines. In terms of fairness, q-FFL is able to reduce the variance of accuracies across devices by 45% on average while maintaining the same overall average accuracy. In terms of efficiency, our distributed method, q-FedAvg, is capable of solving the proposed objective orders-of-magnitude more quickly than other baselines. Finally, while we consider our approaches primarily in the context of federated learning, we also demonstrate that q-FFL can be applied to other related problems such as meta-learning, helping to produce fair initializations across multiple tasks.

4.2 Fair Federated Learning

Our fairness notion in this works follows Definition 1. There are many ways to mathematically evaluate the *uniformity* of the performance. In this work, we mainly use the *variance* of the performance distribution as a measure of uniformity. However, we also explore other uniformity metrics, both empirically and theoretically, in Appendix 4.4.1. We note that a tension exists between the *fairness/uniformity* of the final testing accuracy and the *average* testing accuracy across devices. In general, our goal is to impose more *fairness/uniformity* while maintaining the same (or similar) average accuracy.

Remark 3 (Connections to other fairness definitions)**.** *Definition 1 targets device-level fairness, which has finer granularity than the classical attribute-level fairness such as accuracy parity [315]. We note that in certain cases where devices can be naturally clustered into groups with specific attributes, our definition can be seen as a relaxed version of accuracy parity, in that we optimize for similar but not necessarily identical performance across devices.*

4.2.1 The objective: q-Fair Federated Learning (q-FFL)

A natural idea to achieve fairness as defined in (1) would be to *reweight* the objective—assigning higher weights to devices with poor performance, so that the distribution of accuracies in the network shifts towards more uniformity. Note that this reweighting must be done dynamically, as the performance of the devices depends on the model being trained, which cannot be evaluated a priori. Drawing inspiration from α-fairness, a utility function used in fair resource allocation in wireless networks, we propose the following objective. For given local non-negative cost functions F_k and parameter $q > 0$, we define the q-Fair Federated Learning (q-FFL) objective as:

$$\min_w \; f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)\,, \tag{4.1}$$

where $F_k^{q+1}(\cdot)$ denotes $F_k(\cdot)$ to the power of $(q+1)$. Here, q is a parameter that tunes the amount of fairness we wish to impose. Setting $q = 0$ does not encourage fairness beyond the classical federated learning objective. A larger q means that we emphasize devices with higher local empirical losses, $F_k(w)$, thus imposing more uniformity to the training accuracy distribution and potentially inducing fairness in accordance with Definition 1. Setting $f_q(w)$ with a large enough q reduces to classical minimax fairness [215], as the device with the worst performance (largest loss) will dominate the objective. We note that while the $(q+1)$ term in the denominator in (4.1) may be absorbed in p_k, we include it as it is standard in the α-fairness literature and helps to ease notation. For completeness, we provide additional background on α-fairness in Appendix 4.5.

As mentioned previously, q-FFL generalizes prior work in fair federated learning (AFL) [215], allowing for a flexible tradeoff between fairness and accuracy as parameterized by q. In our theoretical analysis (Appendix 4.4), we provide generalization bounds of q-FFL that generalize the learning bounds of the AFL objective. Moreover, based on

our fairness definition (Definition 1), we theoretically explore how q-FFL results in more *uniform* accuracy distributions with increasing q. Our results suggest that q-FFL is able to impose 'uniformity' of the test accuracy distribution in terms of various metrics such as variance and other geometric and information-theoretic measures.

In our experiments (Section 4.3.2), on both convex and non-convex models, we show that using the q-FFL objective, we can obtain fairer/more uniform solutions for federated datasets in terms of both the training and testing accuracy distributions.

4.2.2 The Solver: FedAvg-Style q-Fair Federated Learning (q-FedAvg)

In developing a functional approach for fair federated learning, it is critical to consider not only what objective to solve but also how to solve such an objective efficiently in a massive distributed network. In this section, we provide methods to solve q-FFL. We start with a simpler method, q-FedSGD, to illustrate our main techniques. We then provide a more efficient counterpart, q-FedAvg, by considering local updating schemes. Our proposed methods closely mirror traditional distributed optimization methods—mini-batch SGD and federated averaging (FedAvg)—but with step-sizes and subproblems carefully chosen in accordance with the q-FFL problem (4.1).

Achieving Variable Levels of Fairness: Tuning q. In devising a method to solve q-FFL (4.1), we begin by noting that it is crucial to first determine how to set q. In practice, q can be tuned based on the desired amount of fairness (with larger q inducing more fairness). As we describe in our experiments (Section 4.3.2), it is therefore common to train a *family of objectives* for different q values so that a practitioner can explore the tradeoff between accuracy and fairness for the application at hand.

One concern with solving such a family of objectives is that it requires step-size tuning for every value of q. In particular, in gradient-based methods, the step-size inversely depends on the Lipschitz constant of the function's gradient, which will change as we change q. This can quickly cause the search space to explode. To overcome this issue, we propose estimating the local Lipschitz constant for the family of q-FFL objectives by using the Lipschitz constant we infer by tuning the step-size (via grid search) on just one q (e.g., $q = 0$). This allows us to dynamically adjust the step-size of our gradient-based optimization method for the q-FFL objective, avoiding manual tuning for each q. In Lemma 1 below we formalize the relation between the Lipschitz constant, L, for $q = 0$ and $q > 0$.

Lemma 1. *If the non-negative function $f(\cdot)$ has a Lipschitz gradient with constant L, then for any $q \geqslant 0$ and at any point w,*

$$L_q(w) = Lf(w)^q + qf(w)^{q-1}\|\nabla f(w)\|^2 \tag{4.2}$$

is an upper-bound for the local Lipschitz constant of the gradient of $\frac{1}{q+1}f^{q+1}(\cdot)$ at point w.

Proof. At any point w, we can compute the Hessian $\nabla^2 \left(\frac{1}{q+1} f^{q+1}(w)\right)$ as:

$$\nabla^2 \left(\frac{1}{q+1} f^{q+1}(w)\right) = q f^{q-1}(w) \underbrace{\nabla f(w) \nabla^T f(w)}_{\preceq \|\nabla f(w)\|^2 \times I} + f^q(w) \underbrace{\nabla^2 f(w)}_{\preceq L \times I}. \tag{4.3}$$

As a result, $\|\nabla^2 \frac{1}{q+1} f^{q+1}(w)\|_2 \leqslant L_q(w) = L f(w)^q + q f(w)^{q-1} \|\nabla f(w)\|^2$. □

A First Approach: q-FedSGD. Our first fair federated learning method, q-FedSGD, is an extension of the well-known federated mini-batch SGD (FedSGD) method [209]. q-FedSGD uses a dynamic step-size instead of the normal fixed step-size of FedSGD. Based on Lemma 1, for each local device k, the upper-bound of the local Lipschitz constant is $L F_k(w)^q + q F_k(w)^{q-1} \|\nabla F_k(w)\|^2$. In each step of q-FedSGD, ∇F_k and F_k on each selected device k are computed at the current iterate and communicated to the central node. This information is used to compute the step-sizes (weights) for combining the updates from each device. The details are summarized in Algorithm 3. Note that q-FedSGD is reduced to FedSGD when $q = 0$. It is also important to note that to run q-FedSGD with different values of q, we only need to estimate L once by tuning the step-size on $q = 0$ and can then reuse it for all values of $q > 0$.

Algorithm 3 q-FedSGD

1: **Input:** K, T, q, $1/L$, w^0, p_k, $k = 1, \cdots, m$
2: **for** $t = 0, \cdots, T-1$ **do**
3: Server selects a subset S_t of K devices at random (each device k is chosen with prob. p_k)
4: Server sends w^t to all selected devices
5: Each selected device k computes:

$$\Delta_k^t = F_k^q(w^t) \nabla F_k(w^t)$$
$$h_k^t = q F_k^{q-1}(w^t) \|\nabla F_k(w^t)\|^2 + L F_k^q(w^t)$$

6: Each selected device k sends Δ_k^t and h_k^t back to the server
7: Server updates w^{t+1} as:

$$w^{t+1} = w^t - \frac{\sum_{k \in S_t} \Delta_k^t}{\sum_{k \in S_t} h_k^t}$$

8: **end for**

Improving Communication-Efficiency: q-FedAvg. In federated settings, communication-efficient schemes using *local* stochastic solvers (such as FedAvg) have been shown to

Algorithm 4 q-FedAvg

1: **Input:** K, E, T, q, $1/L$, η, w^0, p_k, $k = 1, \cdots, m$
2: **for** $t = 0, \cdots, T-1$ **do**
3: Server selects a subset S_t of K devices at random (each device k is chosen with prob. p_k)
4: Server sends w^t to all selected devices
5: Each selected device k updates w^t for E epochs of SGD on F_k with step-size η to obtain $\bar{w}_k^{t+1}$
6: Each selected device k computes:

$$\Delta w_k^t = L(w^t - \bar{w}_k^{t+1})$$
$$\Delta_k^t = F_k^q(w^t)\Delta w_k^t$$
$$h_k^t = qF_k^{q-1}(w^t)\|\Delta w_k^t\|^2 + LF_k^q(w^t)$$

7: Each selected device k sends Δ_k^t and h_k^t back to the server
8: Server updates w^{t+1} as:

$$w^{t+1} = w^t - \frac{\sum_{k \in S_t} \Delta_k^t}{\sum_{k \in S_t} h_k^t}$$

9: **end for**

significantly improve convergence speed [209]. However, when $q > 0$, the F_k^{q+1} term is not an empirical average of the loss over all local samples due to the $q+1$ exponent, preventing the use of local SGD as in FedAvg. To address this, we propose to generalize FedAvg for $q > 0$ using a more sophisticated dynamic weighted averaging scheme. The weights (step-sizes) are inferred from the upper bound of the local Lipschitz constants of the gradients of F_k^{q+1}, similar to q-FedSGD. To extend the local updating technique of FedAvg to the q-FFL objective (4.1), we propose a heuristic where we replace the gradient ∇F_k in the q-FedSGD steps with the local updates that are obtained by running SGD locally on device k. Similarly, q-FedAvg is reduced to FedAvg when $q = 0$. We provide additional details on q-FedAvg in Algorithm 4. As we will see empirically, q-FedAvg can solve q-FFL objective much more efficiently than q-FedSGD due to the local updating heuristic. Finally, recall that as $q \to \infty$ the q-FFL objective recovers that of the AFL. However, we empirically notice that q-FedAvg has a more favorable convergence speed compared to AFL while resulting in similar performance across devices.

4.3 Evaluation

We now present empirical results of the proposed objective, q-FFL, and proposed methods, q-FedAvg and q-FedSGD. We describe our experimental setup in Section 4.3.1. We then

demonstrate the improved fairness of q-FFL in Section 4.3.2, and compare q-FFL with several baseline fairness objectives in Section 4.3.3. Finally, we show the efficiency of q-FedAvg compared with q-FedSGD in Section 4.3.4. All code, data, and experiments are publicly available at `github.com/litian96/fair_flearn`.

4.3.1 Experimental Setup

Federated Datasets. We explore a suite of federated datasets using both convex and non-convex models in our experiments. The datasets are curated from prior work in federated learning [181, 209, 215, 265] as well as recent federated learning benchmarks [43]. In particular, we study: (1) a synthetic dataset using a linear regression classifier, (2) a Vehicle dataset collected from a distributed sensor network [75] with a linear SVM for binary classification, (3) tweet data curated from Sentiment140 [101] (Sent140) with an LSTM classifier for text sentiment analysis, and (4) text data built from *The Complete Works of William Shakespeare* [209] and an RNN to predict the next character. When comparing with AFL, we use the two small benchmark datasets (Fashion MNIST [302] and Adult [34]) studied in [215]. When applying q-FFL to meta-learning, we use the common meta-learning benchmark dataset Omniglot [162]. Full dataset details are given in Appendix 4.6.1.

Implementation. We implement all code in Tensorflow [2], simulating a federated network with one server and m devices, where m is the total number of devices in the dataset (Appendix 4.6.1). We provide full details (including all hyperparameter values) in Appendix 4.6.2.

4.3.2 Fairness of q-FFL

In our first experiments, we verify that the proposed objective q-FFL leads to more fair solutions (Definition 1) for federated data. In Figure 4.2, we compare the final testing accuracy distributions of two objectives ($q = 0$ and a tuned value of $q > 0$) averaged across 5 random shuffles of each dataset. We observe that while the average testing accuracy remains fairly consistent, the objectives with $q > 0$ result in more centered (i.e., fair) testing accuracy distributions with lower variance. In particular, *while maintaining roughly the same average accuracy, q-FFL reduces the variance of accuracies across all devices by 45% on average.* We further report the worst and best 10% testing accuracies and the variance of the final accuracy distributions in Table 4.1. Comparing $q = 0$ and $q > 0$, we see that the average testing accuracy remains almost unchanged with the proposed objective despite significant reductions in variance. We report full results on all uniformity measurements (including variance) in Table 4.5 in the appendix, and show that q-FFL encourages more uniform accuracies under other metrics as well. We observe similar results on training accuracy distributions in Figure 4.6 and Table 4.6, Appendix 4.7. In Table 4.1, the average accuracy is with respect to all data points, not all

devices; however, we observe similar results with respect to devices, as shown in Table 4.7, Appendix 4.7.

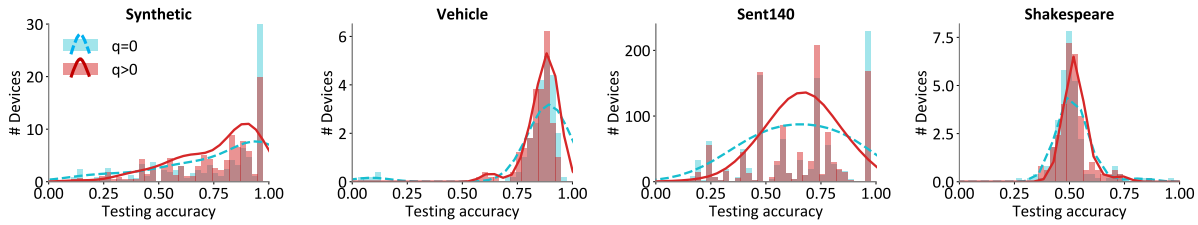


Figure 4.2: q-FFL leads to fairer test accuracy distributions. While the average accuracy remains almost identical (see Table 4.1), by setting $q > 0$, the distributions shift towards the center as low accuracies increase at the cost of potentially decreasing high accuracies on some devices. Setting $q = 0$ corresponds to the original objective (3.1). The selected q values for $q > 0$ on the four datasets, as well as distribution statistics, are also shown in Table 4.1.

Table 4.1: Statistics of the test accuracy distribution for q-FFL. By setting $q > 0$, the accuracy of the worst 10% devices is increased at the cost of possibly decreasing the accuracy of the best 10% devices. While the average accuracy remains similar, the variance of the final accuracy distribution decreases significantly. We provide full results of other uniformity measurements (including variance) in Table 4.5, Appendix 4.7.1, and show that q-FFL encourages more uniform distributions under all metrics.

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**
Synthetic	$q = 0$	80.8 $\pm$ .9	18.8 $\pm$ 5.0	100.0 $\pm$ 0.0	724 $\pm$ 72
	$q = 1$	79.0 $\pm$ 1.2	**31.1** $\pm$ 1.8	100.0 $\pm$ 0.0	**472** $\pm$ 14
Vehicle	$q = 0$	87.3 $\pm$ .5	43.0 $\pm$ 1.0	**95.7** $\pm$ 1.0	291 $\pm$ 18
	$q = 5$	87.7 $\pm$ .7	**69.9** $\pm$ .6	94.0 $\pm$ .9	**48** $\pm$ 5
Sent140	$q = 0$	65.1 $\pm$ 4.8	15.9 $\pm$ 4.9	100.0 $\pm$ 0.0	697 $\pm$ 132
	$q = 1$	66.5 $\pm$ .2	**23.0** $\pm$ 1.4	100.0 $\pm$ 0.0	**509** $\pm$ 30
Shakespeare	$q = 0$	51.1 $\pm$ .3	39.7 $\pm$ 2.8	**72.9** $\pm$ 6.7	82 $\pm$ 41
	$q = .001$	52.1 $\pm$ .3	**42.1** $\pm$ 2.1	69.0 $\pm$ 4.4	**54** $\pm$ 27

Choosing q. As discussed in Section 4.2.2, a natural question is to determine how q should be tuned in the q-FFL objective. Our framework is flexible in that it allows one to choose q to tradeoff between fairness/uniformity and average accuracy. We empirically show that there are a family of q's that can result in variable levels of fairness (and accuracy) on synthetic data in Table 4.11, Appendix 4.7. In general, this value can be tuned based on the data/application at hand and the desired amount of fairness.

Another reasonable approach in practice would be to run Algorithm 4 with multiple q's in parallel to obtain multiple final global models, and then select amongst these based on performance (e.g., accuracy) on the validation data. Rather than using just one optimal q for all devices, for example, each device could pick a device-specific model based on their validation data. We show additional performance improvements with this device-specific strategy in Table 4.12 in Appendix 4.7. Finally, we note that one potential issue is that increasing the value of q may slow the speed of convergence. However, for values of q that result in more fair results on our datasets, we do not observe significant decrease in the convergence speed, as shown in Figure 4.8, Appendix 4.7.

4.3.3 Comparison with Other Objectives

Next, we compare q-FFL with other objectives that are likely to impose fairness in federated networks. One heuristic is to weight each data point equally, which reduces to the original objective in (3.1) (i.e., q-FFL with $q = 0$) and has been investigated in Section 4.3.2. We additionally compare with two alternatives: weighting *devices equally* when sampling devices, and weighting *devices adversarially*, namely, optimizing for the worst-performing device, as proposed in Mohri et al. [215].

Weighting Devices Equally. We compare also q-FFL with uniform sampling schemes and report testing accuracy in Figure 4.3. While the 'weighting each device equally' heuristic tends to outperform our method in *training* accuracy distributions (Figure 4.7 and Table 4.8 in Appendix 4.7), we see that our method produces more fair solutions in terms of *testing* accuracies. One explanation for this is that uniform sampling is a static method and can easily overfit to devices with very few data points, whereas q-FFL will put less weight on a device once its loss becomes small, potentially providing better generalization performance due to its dynamic nature.

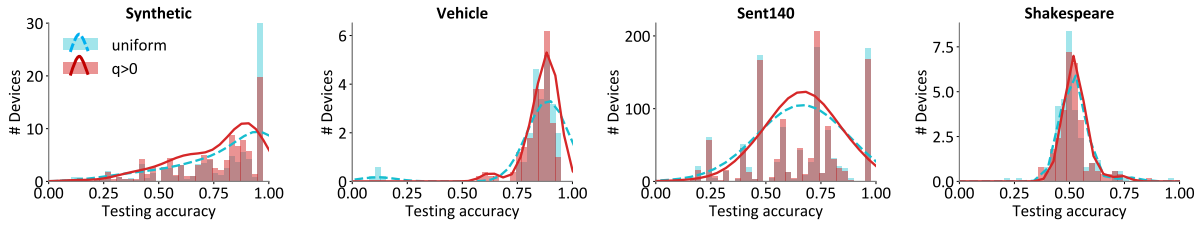


Figure 4.3: q-FFL ($q > 0$) compared with uniform sampling. In terms of testing accuracy, our objective produces more fair solutions than uniform sampling. q-FFL achieves similar average accuracies and more fair solutions.

Weighting Devices Adversarially. We further compare with AFL [215], which is the only work we are aware of that aims to address fairness issues in federated learning. We implement a non-stochastic version of AFL where all devices are selected and updated

each round, and perform grid search on the AFL hyperparameters, γ_w and γ_λ. In order to devise a setup that is as favorable to AFL as possible, we modify Algorithm 4 by sampling all devices and letting each of them run gradient descent at each round. We use the same small datasets (Adult [34] and subsampled Fashion MNIST [302]) and the same logistic regression model as in Mohri et al. [215]. Full details of the implementation and hyperparameters (e.g., values of q_1 and q_2) are provided in the appendix. We note that, as opposed to AFL, q-FFL is flexible depending on the amount of fairness desired, with larger q leading to more accuracy uniformity. As discussed, q-FFL generalizes AFL in this regard, as AFL is equivalent to q-FFL with a large enough q. In Table 4.2, we observe that q-FFL can in fact achieve higher testing accuracy than AFL on the device with the worst performance (i.e., the problem that the AFL was designed to solve) with appropriate q. This also indicates that q-FFL obtains the most fair solutions in certain cases. We also observe that q-FFL converges faster in terms of communication rounds compared with AFL to obtain similar performance (Appendix 4.7), which we speculate is due to the non-smoothness of the AFL objective.

Table 4.2: Our objective compared with weighing devices adversarially (AFL [215]). In order to be favorable to AFL, we use the two small, well-behaved datasets studied in [215]. q-FFL ($q > 0$) outperforms AFL on the worst testing accuracy of both datasets. The tunable parameter q controls how much fairness we would like to achieve—larger q induces less variance. Each accuracy is averaged across 5 runs with different random initializations.

	Adult			**Fashion MNIST**			
Objectives	average (%)	PhD (%)	non-PhD (%)	average (%)	shirt (%)	pullover (%)	T-shirt (%)
q-FFL, q=0	83.2 $\pm$.1	69.9 $\pm$.4	**83.3** $\pm$.1	78.8 $\pm$.2	66.0 $\pm$.7	84.5 $\pm$.8	**85.9** $\pm$.7
AFL	82.5 $\pm$.5	73.0 $\pm$2.2	82.6 $\pm$.5	77.8 $\pm$1.2	71.4 $\pm$4.2	81.0 $\pm$3.6	82.1 $\pm$3.9
q-FFL, q_1>0	82.6 $\pm$.1	**74.1** $\pm$.6	82.7 $\pm$.1	77.8 $\pm$.2	**74.2** $\pm$.3	78.9 $\pm$.4	80.4 $\pm$.6
q-FFL, q_2>q_1	82.3 $\pm$.1	**74.4** $\pm$.9	82.4 $\pm$.1	77.1 $\pm$.4	**74.7** $\pm$.9	77.9 $\pm$.4	78.7 $\pm$.6

4.3.4 Efficiency of the Method q-FedAvg

In this section, we show the efficiency of our proposed distributed solver, q-FedAvg, by comparing Algorithm 4 with its non-local-updating baseline q-FedSGD (Algorithm 3) to solve the same objective (same $q > 0$ as in Table 4.1). At each communication round, we have each method perform the same amount of computation, with q-FedAvg running one epoch of local updates on each selected device while q-FedSGD runs gradient descent with the local training data. In Figure 4.4, q-FedAvg converges faster than q-FedSGD in terms of communication rounds in most cases due to its local updating scheme. The slower convergence of q-FedAvg compared with q-FedSGD on the synthetic dataset may be due to the fact that when local data distributions are highly heterogeneous, local updating schemes may allow local models to move too far away from the initial global model, potentially hurting convergence; see Figure 4.10 in Appendix 4.7 for more details.

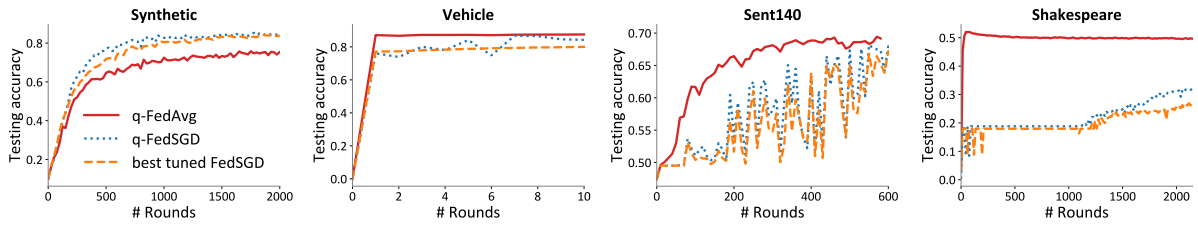


Figure 4.4: For a fixed objective (i.e., q-FFL with the same q), the convergence of q-FedAvg (Algorithm 4), q-FedSGD (Algorithm 3), and FedSGD. For q-FedAvg and q-FedSGD, we tune a best step-size on $q = 0$ and apply that step-size to solve q-FFL with $q > 0$. For q-FedSGD, we tune the step-size directly. We observe that (1) q-FedAvg converges faster in terms of communication rounds; (2) our proposed q-FedSGD solver with a dynamic step-size achieves similar convergence behavior compared with a best-tuned FedSGD.

To demonstrate the optimality of our dynamic step-size strategy in terms of solving q-FFL, we also compare our solver q-FedSGD with FedSGD with a best-tuned step-size. For q-FedSGD, we tune a step-size on $q = 0$ and apply that step-size to solve q-FFL with $q > 0$. q-FedSGD has similar performance with FedSGD, which indicates that (the inverse of) our estimated Lipschitz constant on $q > 0$ is as good as a best tuned fixed step-size. We can reuse this estimation for different q's instead of manually re-tuning it when q changes. We show the full results on other datasets in Appendix 4.7. We note that both proposed methods q-FedAvg and q-FedSGD can be easily integrated into existing implementations of federated learning algorithms such as TensorFlow Federated [1].

4.3.5 Beyond Federated Learning: Applying q-FFL to Meta-Learning

Finally, we generalize the proposed q-FFL objective to other learning tasks beyond federated learning. One natural extension is to apply q-FFL to meta-learning, where each task can be viewed as a device in federated networks. The goal of meta-learning is to learn a model initialization such that it can be quickly adapted to new tasks using limited training samples. However, as the new tasks can be heterogeneous, the performance distribution of the final personalized models may also be non-uniform. Therefore, we aim to learn a better initialization such that it can quickly solve unseen tasks in a fair manner, i.e., reduce the variance of the accuracy distribution of the personalized models.

To achieve this goal, we propose a new method, q-MAML, by combining q-FFL with the popular meta-learning method MAML [92]. In particular, instead of updating the global model in the way described in MAML, we update the global parameters using the gradients of the q-FFL objective $\frac{1}{q+1}F_k^{q+1}(w)$, with weights inferred from Lemma 1. Similarly, q-MAML with $q = 0$ reduces to MAML, and q-MAML with $q \to \infty$ corresponds to MAML with a most 'fair' initialization and a potentially lower average accuracy. The detailed algorithm is summarized in Algorithm 5 in the appendix. We sample 10 tasks at each round during meta-training, and train for 5 iterations of (mini-batch) SGD for personalization on meta-testing tasks. We report test accuracy of personalized models on the meta-testing tasks.

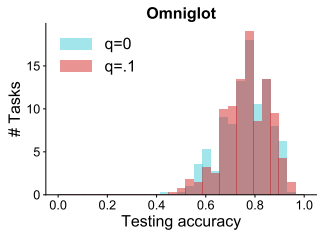


Figure 4.5: q-FFL results in fairer (more centered) initializations for meta-learning tasks.

Table 4.3: Statistics of the accuracy distribution of personalized models using q-MAML. The method with $q = 0$ corresponds to MAML. Similarly, we see that the variance is reduced while the accuracy of the worst 10% devices is increased.

Dataset	**Obj.**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**
Omniglot	$q = 0$	79.1 $\pm$ 9.8	61.2 $\pm$ 3.2	94.0 $\pm$ .5	93 $\pm$ 23
	$q = .1$	79.3 $\pm$ 9.6	**62.5** $\pm$ 5.3	93.8 $\pm$ .9	**86** $\pm$ 28

From Figure 5 and Table 3 above, we observe that q-MAML is able to learn initializations which result in fairer personalized models with lower variance.

4.4 Theoretical Analysis of the Proposed Objective q-FFL

4.4.1 Uniformity Induced by q-FFL

In this section, we theoretically justify that the q-FFL objective can impose more uniformity of the performance/accuracy distribution. As discussed in Section 4.2.1, q-FFL can encourage more fair solutions in terms of several metrics, including (1) the variance of accuracy distribution (smaller variance), (2) the cosine similarity between the accuracy distribution and the all-ones vector $\mathbf{1}$ (larger similarity), and (3) the entropy of the accuracy distribution (larger entropy). We begin by formally defining these fairness notions.

Definition 8 (*Uniformity 1: Variance of the performance distribution*)**.** *We say that the performance distribution of m devices $\{F_1(w), \ldots, F_m(w)\}$ is more uniform under solution w than w' if*

$$\mathbf{Var}(F_1(w), \ldots, F_m(w)) < \mathbf{Var}(F_1(w'), \ldots, F_m(w')).$$

Definition 9 (*Uniformity 2: Cosine similarity between the performance distribution and* $\mathbf{1}$)**.** *We say that the performance distribution of m devices $\{F_1(w), \ldots, F_m(w)\}$ is more uniform under solution w than w' if the cosine similarity between $\{F_1(w), \ldots, F_m(w)\}$ and $\mathbf{1}$ is larger than that between $\{F_1(w'), \ldots, F_m(w')\}$ and $\mathbf{1}$, i.e.,*

$$\frac{\frac{1}{m}\sum_{k=1}^m F_k(w)}{\sqrt{\frac{1}{m}\sum_{k=1}^m F_k^2(w)}} \geqslant \frac{\frac{1}{m}\sum_{k=1}^m F_k(w')}{\sqrt{\frac{1}{m}\sum_{k=1}^m F_k^2(w')}}.$$

Definition 10 (*Uniformity 3: Entropy of performance distribution*)**.** *We say that the performance distribution of m devices $\{F_1(w), \ldots, F_m(w)\}$ is more uniform under solution w than w' if*

$$\widetilde{H}(F(w)) \geqslant \widetilde{H}(F(w')),$$

where $\widetilde{H}(F(w))$ is the entropy of the stochastic vector obtained by normalizing $\{F_1(w), \ldots, F_m(w)\}$, defined as

$$\widetilde{H}(F(w)) := -\sum_{k=1}^m \frac{F_k(w)}{\sum_{k=1}^m F_k(w)} \log\left(\frac{F_k(w)}{\sum_{k=1}^m F_k(w)}\right). \tag{4.4}$$

To enforce uniformity/fairness (defined in Definition 8, 9, and 10), we propose the q-FFL objective to impose more weights on the devices with worse performance. Throughout the proof, for the ease of mathematical exposition, we consider a similar unweighted objective:

$$\min_w \left\{ f_q(w) = \left(\frac{1}{m}\sum_{k=1}^m F_k^{q+1}(w)\right)^{\frac{1}{q+1}} \right\},$$

and we denote w_q^* as the global optimal solution of $\min_w \ f_q(w)$.

We first investigate the special case of $q = 1$ and show that $q = 1$ results in more fair solutions than $q = 0$ based on Definition 8 and Definition 9.

Lemma 2. *$q = 1$ leads to a more fair solution (smaller variance of the model performance distribution) than $q = 0$, i.e., $\mathbf{Var}(F_1(w_1^*), \ldots, F_m(w_1^*)) < \mathbf{Var}(F_1(w_0^*), \ldots, F_m(w_0^*))$.*

Proof. Use the fact that w_1^* is the optimal solution of $\min_w \ f_1(w)$, and w_0^* is the optimal solution of $\min_w \ f_0(w)$, we get

$$\begin{aligned}\frac{\sum_{k=1}^m F_k^2(w_1^*)}{m} - \left(\frac{1}{m}\sum_{i=1}^m F_k(w_1^*)\right)^2 &\leqslant \frac{\sum_{k=1}^m F_k^2(w_0^*)}{m} - \left(\frac{1}{m}\sum_{i=1}^m F_k(w_1^*)\right)^2 \\ &\leqslant \frac{\sum_{k=1}^m F_k^2(w_0^*)}{m} - \left(\frac{1}{m}\sum_{i=1}^m F_k(w_0^*)\right)^2. \end{aligned} \tag{4.5}$$

□

Lemma 3. *$q = 1$ leads to a more fair solution (larger cosine similarity between the performance distribution and $\mathbf{1}$) than $q = 0$, i.e.,*

$$\frac{\frac{1}{m}\sum_{k=1}^m F_k(w_1^*)}{\sqrt{\frac{1}{m}F_k^2(w_1^*)}} \geqslant \frac{\frac{1}{m}\sum_{k=1}^m F_k(w_0^*)}{\sqrt{\frac{1}{m}F_k^2(w_0^*)}}.$$

Proof. As $\frac{1}{m}\sum_{k=1}^m F_k(w_1^*) \geqslant \frac{1}{m}\sum_{k=1}^m F_k(w_0^*)$ and $\frac{1}{m}\sum_{k=1}^m F_k^2(w_1^*) \geqslant \frac{1}{m}\sum_{k=1}^m F_k^2(w_0^*)$, it directly follows that

$$\frac{\frac{1}{m}\sum_{k=1}^m F_k(w_1^*)}{\sqrt{\frac{1}{m}F_k^2(w_1^*)}} \geqslant \frac{\frac{1}{m}\sum_{k=1}^m F_k(w_0^*)}{\sqrt{\frac{1}{m}F_k^2(w_0^*)}}.$$

□

We next provide results based on Definition 10. It states that for arbitrary $q \geqslant 0$, by increasing q for a small amount, we can get more uniform performance distributions defined over higher-orders of the performance.

Lemma 4. *Let $F(w)$ be twice differentiable in w with $\nabla^2 F(w) \succ 0$ (positive definite). The derivative of $\widetilde{H}(F^q(w_p^*))$ with respect to the variable p evaluated at the point $p = q$ is non-negative, i.e.,*

$$\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*))\bigg|_{p=q} \geqslant 0,$$

where $\widetilde{H}(F^q(w_p^*))$ is defined in (4.4)*.*

Proof.

$$\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*))\Big|_{p=q} = -\frac{\partial}{\partial p}\sum_k \frac{F_k^q(w_p^*)}{\sum_\kappa F_\kappa^q(w_p^*)} \ln\left(\frac{F_k^q(w_p^*)}{\sum_\kappa F_\kappa^q(w_p^*)}\right)\Big|_{p=q} \tag{4.6}$$

$$= -\frac{\partial}{\partial p}\sum_k \frac{F_k^q(w_p^*)}{\sum_\kappa F_\kappa^q(w_p^*)} \ln\left(F_k^q(w_p^*)\right)\Big|_{p=q} + \frac{\partial}{\partial p}\ln\sum_\kappa F_\kappa^q(w_p^*)\Big|_{p=q} \tag{4.7}$$

$$= -\sum_k \frac{\left(\frac{\partial}{\partial p}w_p^*\Big|_{p=q}\right)^\top \nabla_w F_k^q(w_q^*)}{\sum_\kappa F_\kappa^q(w_q^*)} \ln\left(F_k^q(w_q^*)\right) - \sum_k \frac{F_k^q(w_q^*)}{\sum_\kappa F_\kappa^q(w_q^*)} \frac{\left(\frac{\partial}{\partial p}w_p^*\Big|_{p=q}\right)^\top \nabla_w F_k^q(w_q^*)}{F_k^q(w_q^*)} \tag{4.8}$$

$$= -\sum_k \frac{\left(\frac{\partial}{\partial p}w_p^*\Big|_{p=q}\right)^\top \nabla_w F_k^q(w_q^*)}{\sum_\kappa F_\kappa^q(w_q^*)} \left(\ln\left(F_k^q(w_q^*)\right)+1\right). \tag{4.9}$$

Now, let us examine $\frac{\partial}{\partial p}w_p^*\Big|_{p=q}$. We know that $\sum_k \nabla_w F_k^p(w_p^*) = 0$ by definition. Taking the derivative with respect to p, we have

$$\sum_k \nabla_w^2 F_k^p(w_p^*)\frac{\partial}{\partial p}w_p^* + \sum_k \left(\ln F_k^p(w_p^*)+1\right)\nabla_w F_k^p(w_p^*) = 0. \tag{4.10}$$

Invoking implicit function theorem,

$$\frac{\partial}{\partial p}w_p^* = -\left(\sum_k \nabla_w^2 F_k^p(w_p^*)\right)^{-1}\sum_k \left(\ln F_k^p(w_p^*)+1\right)\nabla_w F_k^p(w_p^*). \tag{4.11}$$

Plugging $\frac{\partial}{\partial p}w_p^*\Big|_{p=q}$ into (4.9), we get that $\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*))\Big|_{p=q} \geqslant 0$ completing the proof. □

Lemma 4 states that for any p, the performance distribution of $\{F_1^p(w_{p+\epsilon}^*),\ldots,F_m^p(w_{p+\epsilon}^*)\}$ is guaranteed to be more uniform based on Definition 10 than that of $\{F_1^p(w_p^*),\ldots,F_m^p(w_p^*)\}$ for a small enough ϵ. Note that Lemma 4 is different from the existing results on the monotonicity of entropy under the tilt operation, which would imply that $\frac{\partial}{\partial q}\widetilde{H}(F^q(w_p^*)) \leqslant 0$ for all $q \geqslant 0$ (see Beirami et al. [26, Lemma 11]).

Ideally, we would like to prove a result more general than Lemma 4, implying that the distribution $\{F_1^q(w_{p+\epsilon}^*),\ldots,F_m^q(w_{p+\epsilon}^*)\}$ is more uniform than $\{F_1^q(w_p^*),\ldots,F_m^q(w_p^*)\}$ for any p,q and small enough ϵ. We prove this result for the special case of $m=2$ in the following.

Lemma 5. *Let $F(w)$ be twice differentiable in w with $\nabla^2 F(w) > 0$ (positive definite). If $m = 2$, for any $q \in \mathbb{R}^+$, the derivative of $\widetilde{H}(F^q(w_p^*))$ with respect to the variable p is non-negative, i.e.,*

$$\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*)) \geqslant 0,$$

where $\widetilde{H}(F^q(w_p^*))$ is defined in (4.4)*.*

Proof. First, we invoke Lemma 4 to obtain that

$$\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*))\bigg|_{p=q} \geqslant 0. \tag{4.12}$$

Let

$$\theta_q(w) := \frac{F_1^q(w)}{F_1^q(w) + F_2^q(w)}. \tag{4.13}$$

Without loss of generality assume that $\theta_q(w_p^*) \in (0, \frac{1}{2}]$, as we can relabel F_1 and F_2 otherwise. Then, given that $m = 2$, we conclude from (4.12) along with the monotonicity of the binary entropy function in $(0, \frac{1}{2}]$ that

$$\frac{\partial}{\partial p}\theta_q(w_p^*)\bigg|_{p=q} \geqslant 0, \tag{4.14}$$

which in conjunction with (4.13) implies that

$$\frac{\partial}{\partial p}\left(\frac{F_1(w_p^*)}{F_2(w_p^*)}\right)^q\bigg|_{p=q} \geqslant 0. \tag{4.15}$$

Given the monotonicity of x^q with respect to x for all $q > 0$, it can be observed that the above is sufficient to imply that for any $q > 0$,

$$\frac{\partial}{\partial p}\left(\frac{F_1(w_p^*)}{F_2(w_p^*)}\right)^q \geqslant 0. \tag{4.16}$$

Going all of the steps back we would obtain that for all $p > 0$

$$\frac{\partial}{\partial p}\widetilde{H}(F^q(w_p^*)) \geqslant 0. \tag{4.17}$$

This completes the proof of the lemma. □

We conjecture that the statement of Lemma 4 is true for all $q \in \mathbb{R}^+$, which would be equivalent to the statement of Lemma 5 holding true for all $m \in \mathbb{N}$.

Thus far, we provided results that showed that q-FFL promotes fairness in three different senses. Next, we further provide a result on equivalence between the geometric and information-theoretic notions of fairness.

Lemma 6 (*Equivalence between uniformity in entropy and cosine distance*)**.** *q-FFL encourages more uniform performance distributions in the cosine distance sense (Definition 9) if any only if it encourages more uniform performance distributions in the entropy sense (Definition 10), i.e., (a) holds if and only if (b) holds where*
(a) for any $p, q \in \mathbb{R}$, the derivative of $H(F^q(w_p^*))$ with respect to p is non-negative,
(b) for any $0 \leqslant t \leqslant r, 0 \leqslant v \leqslant u$, $\frac{f_t(w_u^*)}{f_r(w_u^*)} \geqslant \frac{f_t(w_v^*)}{f_r(w_v^*)}$.

Proof. Definition 10 is a special case of $H(F^q(w_p^*))$ with $q = 1$. If $\widetilde{H}(F^q(w_p^*))$ increases with p for any p, q, then we are guaranteed to get more fair solutions based on Definition 10. Similarly, Definition 9 is a special case of $\frac{f_t(w_u^*)}{f_r(w_u^*)}$ with $t = 0, r = 1$. If $\frac{f_t(w_u^*)}{f_r(w_u^*)}$ increases with u for any $t \leqslant r$, q-FFL can also obtain more fair solutions under Definition 9.

Next, we show that (a) and (b) are equivalent measures of fairness.

For any $r \geqslant t \geqslant 0$, and any $u \geqslant v \geqslant 0$,

$$\frac{f_t(w_u^*)}{f_r(w_u^*)} \geqslant \frac{f_t(w_v^*)}{f_r(w_v^*)} \iff \ln \frac{f_t(w_u^*)}{f_r(w_u^*)} - \ln \frac{f_t(w_v^*)}{f_r(w_v^*)} \geqslant 0 \tag{4.18}$$

$$\iff \int_v^u \frac{\partial}{\partial \tau} \ln \frac{f_t(w_\tau^*)}{f_r(w_\tau^*)} d\tau \geqslant 0 \tag{4.19}$$

$$\iff \frac{\partial}{\partial p} \ln \frac{f_t(w_p^*)}{f_r(w_p^*)} \geqslant 0, \text{ for any } p \geqslant 0 \tag{4.20}$$

$$\iff \frac{\partial}{\partial p} \ln f_r(w_p^*) - \frac{\partial}{\partial p} \ln f_t(w_p^*) \leqslant 0, \text{ for any } p \geqslant 0 \tag{4.21}$$

$$\iff \int_t^r \frac{\partial^2}{\partial p \partial q} \ln f_q(w_p^*) dq \leqslant 0 \text{ for any } p, q \geqslant 0 \tag{4.22}$$

$$\iff \frac{\partial^2}{\partial p \partial q} \ln f_q(w_p^*) \leqslant 0, \text{ for any } p, q \geqslant 0 \tag{4.23}$$

$$\iff \frac{\partial}{\partial p} \widetilde{H}(F^q(w_p^*)) \geqslant 0, \text{ for any } p, q \geqslant 0. \tag{4.24}$$

The last inequality is obtained using the fact that by taking the derivative of $\ln f_q(w_p^*)$ with respect to q, we get $-\widetilde{H}(F^q(w_p^*))$. □

Discussions. We give geometric (Definition 9) and information-theoretic (Definition 10) interpretations of our uniformity/fairness notion and provide uniformity guarantees under the q-FFL objective in some cases (Lemma 2, Lemma 3, and Lemma 4). We reveal interesting relations between the geometric and information-theoretic interpretations in Lemma 6. Future work would be to gain further understandings for more general cases indicated in Lemma 6.

4.4.2 Generalization Bounds

In this section, we first describe the setup we consider in more detail, and then provide generalization bounds of q-FFL. One benefit of q-FFL is that it allows for a flexible tradeoff between fairness and accuracy, which generalizes AFL (a special case of q-FFL with $q \to \infty$). We also provide learning bounds that generalize the bounds of the AFL objective, as described below.

Suppose the service provider is interested in minimizing the loss over a distributed network of devices, with possibly unknown weights on each device:

$$L_\lambda(h) = \sum_{k=1}^{m} \lambda_k \mathbb{E}_{(x,y)\sim D_k}[l(h(x), y)], \tag{4.25}$$

where λ is in a probability simplex Λ, m is the total number of devices, D_k is the local data distribution for device k, h is the hypothesis function, and l is the loss. We use $\hat{L}_\lambda(h)$ to denote the empirical loss:

$$\hat{L}_\lambda(h) = \sum_{k=1}^{m} \frac{\lambda_k}{n_k} \sum_{j=1}^{n_k} l(h(x_{k,j}), y_{k,j}), \tag{4.26}$$

where n_k is the number of local samples on device k and $(x_{k,j}, y_{k,j}) \sim D_k$.

We consider a slightly different, unweighted version of q-FFL:

$$\min_w \; f_q(w) = \frac{1}{m} \sum_{k=1}^{m} F_k^{q+1}(w)\,, \tag{4.27}$$

which is equivalent to minimizing the empirical loss

$$\tilde{L}_q(h) = \max_{\nu, \|\nu\|_p \leqslant 1} \sum_{k=1}^{m} \frac{\nu_i}{n_k} \sum_{j=1}^{n_k} l(h(x_{k,j}), y_{k,j}), \tag{4.28}$$

where $\frac{1}{p} + \frac{1}{q+1} = 1$ $(p \geqslant 1, q \geqslant 0)$.

Lemma 7 (*Generalization bounds of q-FFL for a specific λ*)**.** *Assume that the loss l is bounded by $M > 0$ and the numbers of local samples are $(n_1, \cdots, n_m)$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for any $\lambda \in \Lambda, h \in H$:*

$$L_\lambda(h) \leqslant A_q(\lambda)\tilde{L}_q(h) + \mathbb{E}\left[\max_{h\in H} L_\lambda(h) - \hat{L}_\lambda(h)\right] + M\sqrt{\sum_{k=1}^{m} \frac{\lambda_k^2}{2n_k} \log\frac{1}{\delta}},$$

where $A_q(\lambda) = \|\lambda\|_p$, and $1/p + 1/(q+1) = 1$.

Proof. Similar to the proof in Mohri et al. [215], for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$ for any $\lambda \in \Lambda, h \in H$:

$$L_\lambda(h) \leqslant \hat{L}_\lambda(h) + \mathbb{E}\left[\max_{h \in H} L_\lambda(h) - \hat{L}_\lambda(h)\right] + M\sqrt{\sum_{k=1}^{m} \frac{\lambda_k^2}{2n_k} \log \frac{1}{\delta}}. \tag{4.29}$$

Denote the empirical loss on device k $\frac{1}{n_k}\sum_{j=1}^{n_k} l(h(x_{k,j}), y_{k,j})$ as F_k. From Hölder's inequality, we have

$$\hat{L}_\lambda(h) = \sum_{k=1}^{m} \lambda_k F_k \leqslant \left(\sum_{k=1}^{m} \lambda_k^p\right)^{\frac{1}{p}} \left(\sum_{k=1}^{m} F_k^{q+1}\right)^{\frac{1}{q+1}} = A_q(\lambda)\tilde{L}_q(h), \ \frac{1}{p} + \frac{1}{q+1} = 1.$$

Plugging $\hat{L}_\lambda(h) \leqslant A_q(\lambda)\tilde{L}_q(h)$ into (4.29) yields the results. □

Theorem 4 (*Generalization bounds of q-FFL for any λ*)**.** *Assume that the loss l is bounded by $M > 0$ and the number of local samples is $(n_1, \cdots, n_m)$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for any $\lambda \in \Lambda, h \in H$:*

$$L_\lambda(h) \leqslant \max_{\lambda \in \Lambda} \left(A_q(\lambda)\right) \tilde{L}_q(h) + \max_{\lambda \in \Lambda} \left(\mathbb{E}\left[\max_{h \in H} L_\lambda(h) - \hat{L}_\lambda(h)\right] + M\sqrt{\sum_{k=1}^{m} \frac{\lambda_k^2}{2n_k} \log \frac{1}{\delta}}\right),$$

where $A_q(\lambda) = \|\lambda\|_p$, and $1/p + 1/(q+1) = 1$.

Proof. This directly follows from Lemma 7, by taking the maximum over all possible λ's in Λ. □

Discussions. From Lemma 7, letting $\lambda = \left(\frac{1}{m}, \cdots, \frac{1}{m}\right)$ and $q \to \infty$, we recover the generalization bounds in AFL [215]. In that sense, our generalization results extend those of AFL's. In addition, it is not straightforward to derive an optimal q with the tightest generalization bound from Lemma 7 and Theorem 4. In practice, our proposed method q-FedAvg allows us to tune a family of q's by re-using the step-sizes.

4.5 α-fairness and q-FFL

While it is natural to consider the α-fairness framework for machine learning, we are unaware of any work that uses α-fairness to modify machine learning training objectives. We provide additional details on the framework below; for further background on α-fairness and fairness in resource allocation more generally, we defer the reader to Mo and Walrand [214], Shi et al. [256].

α-fairness [164, 214] is a popular fairness metric widely-used in resource allocation problems. The framework defines a family of overall utility functions that can be derived

by summing up the following function of the individual utilities of the users in the network:

$$U_\alpha(x) = \begin{cases} \ln(x), & \text{if } \alpha = 1 \\ \frac{1}{1-\alpha} x^{1-\alpha}, & \text{if } \alpha \geqslant 0, \alpha \neq 1\,. \end{cases}$$

Here $U_\alpha(x)$ represents the individual utility of some specific user given x allocated resources (e.g., bandwidth). The goal is to find a resource allocation strategy to maximize the sum of the individual utilities. This family of functions includes a wide range of popular fair resource allocation strategies. In particular, the above function represents zero fairness with $\alpha = 0$, proportional fairness [150] with $\alpha = 1$, harmonic mean fairness [62] with $\alpha = 2$, and max-min fairness [234] with $\alpha = +\infty$.

Note that in federated learning, we are dealing with costs and not utilities. Thus, max-min in resource allocation corresponds to min-max in our setting. With this analogy, it is clear that in our proposed objective q-FFL (4.1), the case where $q = +\infty$ corresponds to min-max fairness since it is optimizing for the worst-performing device, similar to what was proposed in Mohri et al. [215]. Also, $q = 0$ corresponds to zero fairness, which reduces to the original FedAvg objective (3.1). In resource allocation problems, α can be tuned for tradeoffs between fairness and system efficiency. In federated settings, q can be tuned based on the desired level of fairness (e.g., desired variance of accuracy distributions) and other performance metrics such as the overall accuracy. For instance, in Table 4.2 in Section 4.3.3, we demonstrate on two datasets that as q increases, the overall average accuracy decreases slightly while the worst accuracies are increased significantly and the variance of the accuracy distribution decreases.

4.6 Experimental Details

4.6.1 Datasets and Models

We provide full details on the datasets and models used in our experiments. The statistics of four federated datasets used in federated learning (as opposed to meta-learning) experiments are summarized in Table 4.4. We report the total number of devices, the total number of samples, and mean and deviation in the sizes of total data points on each device. Additional details on the datasets and models are described below.

- **Synthetic:** We follow a similar set up as that in Shamir et al. [253] and impose additional heterogeneity. The model is $y = argmax(\text{softmax}(Wx + b))$, $x \in \mathbb{R}^{60}, W \in \mathbb{R}^{10\times 60}, b \in \mathbb{R}^{10}$, and the goal is to learn a global W and b. Samples (X_k, Y_k) and local models on each device k satisfies $W_k \sim \mathcal{N}(u_k, 1)$, $b_k \sim \mathcal{N}(u_k, 1)$, $u_k \sim \mathcal{N}(0, 1)$; $x_k \sim \mathcal{N}(v_k, \Sigma)$, where the covariance matrix Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. Each element in v_k is drawn from $\mathcal{N}(B_k, 1)$, $B_k \sim \mathcal{N}(0, 1)$. There are 100 devices in total and the number of samples on each devices follows a power law.

- **Vehicle**[2]**:** We use the same Vehicle Sensor (Vehicle) dataset as Smith et al. [265], modelling each sensor as a device. This dataset consists of acoustic, seismic, and infrared sensor data collected from a distributed network of 23 sensors [75]. Each sample has a 100-dimension feature and a binary label. We train a linear SVM to predict between AAV-type and DW-type vehicles. We tune the hyperparameters in SVM and report the best configuration.
- **Sent140:** This dataset is a collection of tweets curated from 1,101 accounts from Sentiment140 [101] (Sent140) where each Twitter account corresponds to a device. The task is text sentiment analysis which we model as a binary classification problem. The model takes as input a 25-word sequence, embeds each word into a 300-dimensional space using pretrained Glove [228], and outputs a binary label after two LSTM layers and one densely-connected layer.
- **Shakespeare:** This dataset is built from *The Complete Works of William Shakespeare* [209]. Each speaking role in the plays is associated with a device. We subsample 31 speaking roles to train a deep language model for next character prediction. The model takes as input an 80-character sequence, embeds each character into a learnt 8-dimensional space, and outputs one character after two LSTM layers and one densely-connected layer.
- **Omniglot:** The Omniglot dataset [162] consists of 1,623 characters from 50 different alphabets. We create 300 meta-training tasks from the first 1,200 characters, and 100 meta-testing tasks from the last 423 characters. Each task is a 5-class classification problem where each character forms a class. The model is a convolutional neural network with two convolution layers and two fully-connected layers.

Table 4.4: Statistics of federated datasets

Dataset	**Devices**	**Samples**	**Samples/device**	
			mean	stdev
Synthetic	100	12,697	127	73
Vehicle	23	43,695	1,899	349
Sent140	1,101	58,170	53	32
Shakespeare	31	116,214	3,749	6,912

4.6.2 Implementation Details

Machines. We simulate the federated setting (one server and m devices) on a server with 2 Intel® Xeon® E5-2650 v4 CPUs and 8 NVidia® 1080Ti GPUs.

Software. We implement all code in TensorFlow [2] Version 1.10.1.
Please see `github.com/litian96/fair_flearn` for full details.

[2] `http://www.ecs.umass.edu/~mduarte/Software.html`

Algorithm 5 q-FFL applied to MAML: q-MAML

Input: K, T, η, w^0, N, p_k, $k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Sample a batch of S_t ($|S_t| = K$) tasks randomly (each task k is chosen with probability p_k)
  Send w^t to all sampled tasks
  Each task $k \in S_t$ samples data D_k from the training set and D'_k from the testing set, and computes updated parameters on D_k: $w_k^t = w^t - \eta \nabla F_k(w^t)$
  Each task $k \in S_t$ computes the gradients $\nabla F_k(w_k^t)$ on D'_k
  Each task $k \in S_t$ computes:

$$\Delta_k^t = F_k^q(w_k^t) \nabla F_k(w_k^t)$$
$$h_k^t = q F_k^{q-1}(w_k^t) \|\nabla F_k(w_k^t)\|^2 + L F_k^q(w_k^t)$$

  w^{t+1} is updated as:

$$w^{t+1} = w^t - \frac{\sum_{k \in S_t} \Delta_k^t}{\sum_{k \in S_t} h_k^t}$$

end for

Hyperparameters. We randomly split data on each local device into 80% training set, 10% test set, and 10% validation set. We tune a best q from $\{0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10, 15\}$ on the validation set and report accuracy distributions on the testing set. We pick up the q value where the variance decreases the most, while the overall average accuracy change (compared with the $q = 0$ case) is within 1%. For each dataset, we repeat this process for five randomly selected train/test/validation splits, and report the mean and standard deviation across these five runs where applicable. For Synthetic, Vehicle, Sent140, and Shakespeare, optimal q values are 1, 5, 1, and 0.001, respectively. For all datasets, we randomly sample 10 devices each round. We tune the learning rate and batch size on FedAvg and use the same learning rate and batch size for all q-FedAvg experiments of that dataset. The learning rates for Synthetic, Vehicle, Sent140, and Shakespeare are 0.1, 0.01, 0.03, and 0.8, respectively. The batch sizes for Synthetic, Vehicle, Sent140, and Shakespeare are 10, 64, 32, and 10. The number of local epochs E is fixed to be 1 for both FedAvg and q-FedAvg regardless of the values of q.

In comparing q-FedAvg's efficiency with q-FedSGD, we also tune a best learning rate for q-FedSGD methods on $q = 0$. For each comparison, we fix devices selected and mini-batch orders across all runs. We stop training when the training loss $F(w)$ does not decrease for 10 rounds. When running AFL methods, we search for a best γ_w and γ_λ such that AFL achieves the highest testing accuracy on the device with the highest loss within a fixed number of rounds. For Adult, we use $\gamma_w = 0.1$ and $\gamma_\lambda = 0.1$; for Fashion MNIST, we use $\gamma_w = 0.001$ and $\gamma_\lambda = 0.01$. We use the same γ_w as step-sizes for q-FedAvg on Adult and

Fashion MNIST. In Table 4.2, $q_1 = 0.01, q_2 = 2$ for q-FFL on Adult and $q_1 = 5, q_2 = 15$ for q-FFL on Fashion MNIST. Similarly, the number of local epochs is fixed to 1 whenever we perform local updates.

4.7 Full Experiments

4.7.1 Full Results of Previous Experiments

Fairness of q-FFL Under All Uniformity Metrics. We demonstrate the fairness of q-FFL in Table 4.1 in terms of variance. Here, we report similar results in terms of other uniformity measures (the last two columns).

Table 4.5: Full statistics of the test accuracy distribution for q-FFL. q-FFL increases the accuracy of the worst 10% devices without decreasing the average accuracies. We see that q-FFL encourages more uniform distributions under all uniformity metrics defined in Appendix 4.4.2: (1) the variance of the accuracy distribution (Definition 8), (2) the cosine similarity/geometric angle between the accuracy distribution and the all-ones vector $\mathbf{1}$ (Definition 9), and (3) the KL-divergence between the normalized accuracy vector $\boldsymbol{a}$ and the uniform distribution $\boldsymbol{u}$, which can be directly translated to the entropy of $\boldsymbol{a}$ (Definition 10) .

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**	**Angle** ($^\circ$)	**KL($\boldsymbol{a}\|\boldsymbol{u}$)**
Synthetic	$q = 0$	80.8 $\pm$ .9	18.8 $\pm$ 5.0	100.0 $\pm$ 0.0	724 $\pm$ 72	19.5 $\pm$ 1.1	.083 $\pm$ .013
	$q = 1$	79.0 $\pm$ 1.2	**31.1** $\pm$ 1.8	100.0 $\pm$ 0.0	**472** $\pm$ 14	**16.0** $\pm$ .5	**.049** $\pm$ .003
Vehicle	$q = 0$	87.3 $\pm$ .5	43.0 $\pm$ 1.0	**95.7** $\pm$ 1.0	291 $\pm$ 18	11.3 $\pm$ .3	.031 $\pm$ .003
	$q = 5$	87.7 $\pm$ .7	**69.9** $\pm$ .6	94.0 $\pm$ .9	**48** $\pm$ 5	**4.6** $\pm$ .2	**.003** $\pm$ .000
Sent140	$q = 0$	65.1 $\pm$ 4.8	15.9 $\pm$ 4.9	100.0 $\pm$ 0.0	697 $\pm$ 132	22.4 $\pm$ 3.3	.104 $\pm$ .034
	$q = 1$	66.5 $\pm$ .2	**23.0** $\pm$ 1.4	100.0 $\pm$ 0.0	**509** $\pm$ 30	**18.8** $\pm$ .5	**.067** $\pm$ .006
Shakespeare	$q = 0$	51.1 $\pm$ .3	39.7 $\pm$ 2.8	**72.9** $\pm$ 6.7	82 $\pm$ 41	9.8 $\pm$ 2.7	.014 $\pm$ .006
	$q = .001$	52.1 $\pm$ .3	**42.1** $\pm$ 2.1	69.0 $\pm$ 4.4	**54** $\pm$ 27	**7.9** $\pm$ 2.3	**.009** $\pm$ .05

Fairness of q-FFL w.r.t. Training Accuracy. The empirical results in Section 4.3 are with respect to testing accuracy. As a sanity check, we show that q-FFL also results in more fair training accuracy distributions in Figure 4.6 and Table 4.6.

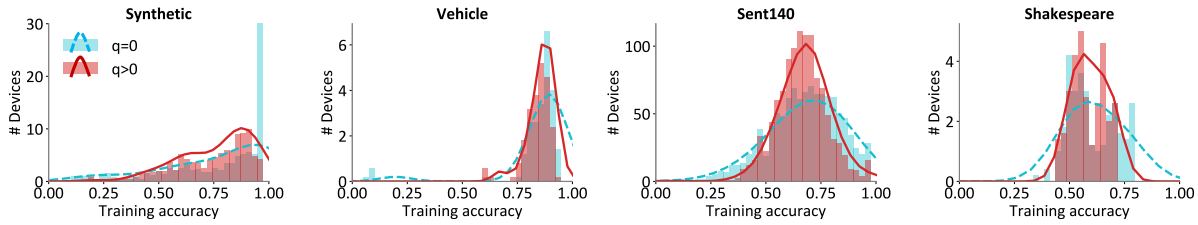


Figure 4.6: q-FFL ($q > 0$) results in more centered (i.e., fair) *training* accuracy distributions across devices without sacrificing the average accuracy.

Table 4.6: q-FFL results in more fair training accuracy distributions in terms of all uniformity measurements—(a) the accuracy variance, (b) the cosine similarity (i.e., angle) between the accuracy distribution and the all-ones vector $\mathbf{1}$, and (c) the KL divergence between the normalized accuracy $\boldsymbol{a}$ and uniform distribution $\boldsymbol{u}$.

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**	**Angle** (°)	**KL($\boldsymbol{a}\|\boldsymbol{u}$)**
Synthetic	$q = 0$	81.7 ± .3	23.6 ± 1.1	100.0 ± .0	597 ± 10	17.5 ± .3	.061 ± .002
	$q = 1$	78.9 ± .2	**41.8** ± 1.0	96.8 ± .5	**292** ± 11	**12.5** ± .2	**.027** ± .001
Vehicle	$q = 0$	87.5 ± .2	49.5 ± 10.2	**94.9** ± .7	237 ± 97	10.2 ± 2.4	.025 ± .011
	$q = 5$	87.8 ± .5	**71.3** ± 2.2	93.1 ± 1.4	**37** ± 12	**4.0** ± .7	**.003** ± .001
Sent140	$q = 0$	69.8 ± .8	36.9 ± 3.1	**94.4** ± 1.1	278 ± 44	13.6 ± 1.1	.032 ± .006
	$q = 1$	68.2 ± .6	**46.0** ± .3	88.8 ± .8	**143** ± 4	**10.0** ± .1	**.017** ± .000
Shakespeare	$q = 0$	72.7 ± .8	46.4 ± 1.4	**79.7** ± .9	116 ± 8	9.9 ± .3	.015 ± .001
	$q = .001$	66.7 ± 1.2	**48.0** ± .4	71.2 ± 1.9	**56** ± 9	**7.1** ± .5	**.008** ± .001

Average Testing Accuracy w.r.t. Devices. In Section 4.3.2, we show that q-FFL leads to more fair accuracy distributions while maintaining approximately the same testing accuracies. Note that we report average testing accuracy with respect to *all data points* in Table 4.1. However, we observe similar results on average accuracy with respect to *all devices* between $q = 0$ and $q > 0$ objectives, as shown in Table 4.7. This indicates that q-FFL can reduce the variance of the accuracy distribution without sacrificing the average accuracy over devices or over data points.

Table 4.7: Average testing accuracy under q-FFL objectives. We show that the resulting solutions of $q = 0$ and $q > 0$ objectives have approximately the same average accuracies both with respect to all data points and with respect to all devices.

Dataset	**Objective**	**Accuracy w.r.t. Data Points** (%)	**Accuracy w.r.t. Devices** (%)
Synthetic	$q = 0$	80.8 $\pm$ .9	77.3 $\pm$ .6
	$q = 1$	79.0 $\pm$ 1.2	76.3 $\pm$ 1.7
Vehicle	$q = 0$	87.3 $\pm$ .5	85.6 $\pm$ .4
	$q = 5$	87.7 $\pm$ .7	86.5 $\pm$ .7
Sent140	$q = 0$	65.1 $\pm$ 4.8	64.6 $\pm$ 4.5
	$q = 1$	66.5 $\pm$ .2	66.2 $\pm$ .2
Shakespeare	$q = 0$	51.1 $\pm$ .3	61.4 $\pm$ 2.7
	$q = .001$	52.1 $\pm$ .3	60.0 $\pm$ .5

Comparison with Uniform Sampling Devices. In Figure 4.7 and Table 4.8, we show that in terms of training accuracies, the uniform sampling heuristic may outperform q-FFL (as opposed to the testing accuracy results in Section 4.3). We suspect that this is because the uniform sampling baseline is a static method and is likely to overfit to those devices with few samples. In additional to Figure 4.3 in Section 4.3.3, we also report the average testing accuracy with respect to data points, best 10%, worst 10% accuracies, and the variance (along with two other uniformity measures) in Table 4.9.

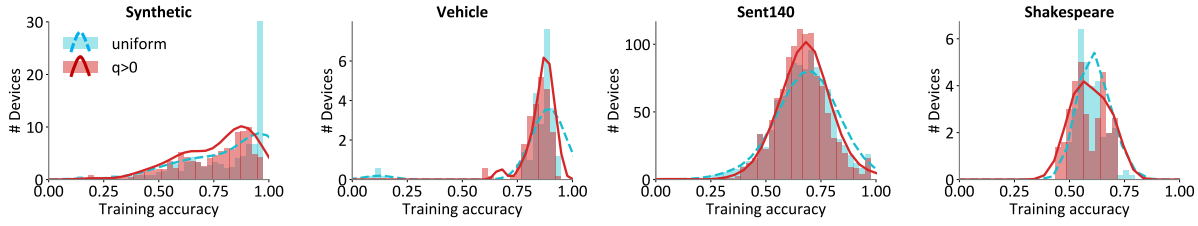


Figure 4.7: q-FFL ($q > 0$) compared with uniform sampling in training accuracy. We see that on some datasets uniform sampling has higher (and more fair) training accuracies due to the fact that it is overfitting to devices with few samples.

Table 4.8: More statistics comparing the uniform sampling objective with q-FFL in terms of training accuracies. We observe that uniform sampling could result in more fair *training* accuracy distributions with smaller variance in some cases.

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**	**Angle** (°)	**KL($a\|u$)**
Synthetic	uniform	83.5 ± .2	**42.6** ± 1.4	**100.0** ± .0	366 ± 17	13.4 ± .3	.031 ± .002
	$q = 1$	78.9 ± .2	41.8 ± 1.0	96.8 ± .5	**292** ± 11	**12.5** ± .2	**.027** ± .001
Vehicle	uniform	87.3 ± .3	46.6 ± .8	**94.8** ± .5	261 ± 10	10.7 ± .2	.027 ± .001
	$q = 5$	87.8 ± .5	**71.3** ± 2.2	93.1 ± 1.4	**37** ± 12	**4.0** ± .7	**.003** ± .001
Sent140	uniform	69.1 ± .5	42.2 ± 1.1	**91.0** ± 1.3	188 ± 19	11.3 ± .5	.022 ± .002
	$q = 1$	68.2 ± .6	**46.0** ± .3	88.8 ± .8	**143** ± 4	**10.0** ± .1	**.017** ± .000
Shakespeare	uniform	57.7 ± 1.5	**54.1** ± 1.7	**72.4** ± 3.2	**32** ± 7	**5.2** ± .5	**.004** ± .001
	$q = .001$	66.7 ± 1.2	48.0 ± .4	71.2 ± 1.9	56 ± 9	7.1 ± .5	.008 ± .001

Table 4.9: More statistics showing more fair solutions induced by q-FFL compared with the uniform sampling baseline in terms of *test* accuracies. Again, we observe that under q-FFL, the testing accuracy of the worst 10% devices tends to increase compared with uniform sampling, and the variance of the final testing accuracies is smaller. Similarly, q-FFL is also more fair than uniform sampling in terms of other uniformity metrics.

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**	**Angle** (°)	**KL($a\|u$)**
Synthetic	uniform	82.2 ± 1.1	30.0 ± .4	100.0 ± .0	525 ± 47	**15.6** ± .8	**.048** ± .007
	$q = 1$	79.0 ± 1.2	**31.1** ± 1.8	100.0 ± 0.0	**472** ± 14	16.0 ± .5	.049 ± .003
Vehicle	uniform	86.8 ± .3	45.4 ± .3	**95.4** ± .7	267 ± 7	10.8 ± .1	.028 ± .001
	$q = 5$	87.7 ± 0.7	**69.9** ± .6	94.0 ± .9	**48** ± 5	**4.6** ± .2	**.003** ± .000
Sent140	uniform	66.6 ± 2.6	21.1 ± 1.9	100.0 ± 0.0	560 ± 19	19.8 ± .7	.076 ± .006
	$q = 1$	66.5 ± .2	**23.0** ± 1.4	100.0 ± 0.0	**509** ± 30	**18.8** ± .5	**.067** ± .006
Shakespeare	uniform	50.9 ± .4	41.0 ± 3.7	**70.6** ± 5.4	71 ± 38	9.1 ± 2.8	.012 ± .006
	$q = .001$	52.1 ± .3	**42.1** ± 2.1	69.0 ± 4.4	**54** ± 27	**7.9** ± 2.3	**.009** ± .05

4.7.2 Additional Experiments

Effects of Data Heterogeneity and The Number of Devices on Unfairness. To study how data heterogeneity and the total number of devices affect unfairness in a more direct way, we investigate into a set of synthetic datasets where we can quantify the degree of heterogeneity. The results are shown in Table 4.10 below. We generate three synthetic datasets following the process described in Appendix 4.6.1, but with different parameters to control heterogeneity. In particular, we generate an IID data— Synthetic (IID) by setting the same W and b on all devices and setting the samples $x_k \sim \mathcal{N}(0, 1)$ for any device k. We instantiate two non-identically distributed datasets (Synthetic (1, 1) and Synthetic (2, 2)) from Synthetic (α, β) where $u_k \sim \mathcal{N}(0, \alpha)$ and $B_k \sim \mathcal{N}(0, \beta)$. Recall that α, β allows to precisely manipulate the degree of heterogeneity with larger α, β values indicating more statistical heterogeneity. Therefore, from top to bottom in Table 4.10, data are more heterogeneous. For each dataset, we further create two variants with different number of participating devices. We see that as data become more heterogeneous and as the number of devices in the network increases, the accuracy distribution tends to be less uniform.

Table 4.10: Effects of data heterogeneity and the number of devices on unfairness. For a fixed number of devices, as data heterogeneity increases from top to bottom, the accuracy distributions become less uniform (with larger variance) for both $q = 0$ and $q > 0$. Within each dataset, the decreasing number of devices results in a more uniform accuracy distribution. In all scenarios (except on IID data), setting $q > 0$ helps to encourage more fair solutions.

Dataset		Objective	Average	Worst 10%	Best 10%	Variance
Synthetic (IID)	100 devices	$q = 0$	89.2 $\pm$ .6	70.9 $\pm$ 3	100.0 $\pm$ 0	85 $\pm$ 15
		$q = 1$	89.0 $\pm$ .5	70.3 $\pm$ 3	100.0 $\pm$ 0	88 $\pm$ 19
	50 devices	$q = 0$	87.1 $\pm$ 1.5	66.5 $\pm$ 3	100.0 $\pm$ 0	107 $\pm$ 14
		$q = 1$	86.8 $\pm$ 0.8	66.5 $\pm$ 2	100.0 $\pm$ 0	109 $\pm$ 13
Synthetic (1, 1)	100 devices	$q = 0$	83.0 $\pm$ .9	36.8 $\pm$ 2	100.0 $\pm$ 0	452 $\pm$ 22
		$q = 1$	82.7 $\pm$ 1.3	**43.5** $\pm$ 5	100.0 $\pm$ 0	**362** $\pm$ 58
	50 devices	$q = 0$	84.5 $\pm$ .3	43.3 $\pm$ 2	100.0 $\pm$ 0	370 $\pm$ 37
		$q = 1$	85.1 $\pm$ .8	**47.3** $\pm$ 3	100.0 $\pm$ 0	**317** $\pm$ 41
Synthetic (2, 2)	100 devices	$q = 0$	82.6 $\pm$ 1.1	25.5 $\pm$ 8	100.0 $\pm$ 0	618 $\pm$ 117
		$q = 1$	82.2 $\pm$ 0.7	**31.9** $\pm$ 6	100.0 $\pm$ 0	**484** $\pm$ 79
	50 devices	$q = 0$	85.9 $\pm$ 1.0	36.8 $\pm$ 7	100.0 $\pm$ 0	421 $\pm$ 85
		$q = 1$	85.9 $\pm$ 1.4	**39.1** $\pm$ 6	100.0 $\pm$ 0	**396** $\pm$ 76

A Family of q's Results in Variable Levels of Fairness. In Table 4.11, we show the accuracy distribution statistics of using a family of q's on synthetic data. Our objective and methods are not sensitive to any particular q since all $q > 0$ values can lead to more fair solutions compared with $q = 0$. In our experiments in Section 4.3, we report the results using the q values selected following the protocol described in Appendix 4.6.2.

Table 4.11: Test accuracy statistics of using a family of q's on synthetic data. We show results with q's selected from our candidate set $\{0.001, 0.01, 0.1, 1, 2, 5, 10, 15\}$. q-FFL allows for a more flexible tradeoff between fairness and accuracy. A larger q results in more fairness (smaller variance), but potentially lower accuracy. Similarly, a larger q imposes more uniformity in terms of other metrics—(a) the cosine similarity/angle between the accuracy distribution and the all-ones vector $\mathbf{1}$, and (b) the KL divergence between the normalized accuracy $\boldsymbol{a}$ and a uniform distribution $\boldsymbol{u}$.

Dataset	Objective	Average (%)	Worst 10% (%)	Best 10% (%)	Variance	Angle (°)	KL($\boldsymbol{a}\|\boldsymbol{u}$)
Synthetic	q=0	80.8 ± .9	18.8 ± 5.0	100.0 ± 0.0	724 ± 72	19.5 ± 1.1	.083 ± .013
	q= 0.1	81.1 ± 0.8	22.1 ± .8	100.0 ± 0.0	666 ± 56	18.4 ± .8	.070 ± .009
	q=1	79.0 ± 1.2	31.1 ± 1.8	100.0 ± 0.0	472 ± 14	16.0 ± .5	.049 ± .003
	q=2	74.7 ± 1.3	32.2 ± 2.1	99.9 ± .2	410 ± 23	15.6 ± 0.7	.044 ± .005
	q=5	67.2 ± 0.9	30.0 ± 4.8	94.3 ± 1.4	369 ± 51	16.3 ± 1.2	.048 ± .010

Device-Specific q. In these experiments, we explore a device-specific strategy for selecting q in q-FFL. We solve q-FFL with $q \in \{0, 0.001, 0.01, 0.1, 1, 2, 5, 10\}$ in parallel. After training, each device selects the best resulting model based on the validation data and tests the performance of the model using the testing set. We report the results in terms of testing accuracy in Table 4.12. Interestingly, using this device-specific strategy the average accuracy in fact increases while the variance of accuracies is reduced, in comparison with $q = 0$. We note that this strategy does induce more local computation and additional communication load at each round. However, it does not increase the number of communication rounds if run in parallel.

Table 4.12: Effects of running q-FFL with several q's in parallel. We train multiple global models (corresponding to different q's) independently in the network. After the training finishes, each device picks up a best, device-specific model based on the performance (accuracy) on the validation data. While this adds additional local computation and more communication load per round, the device-specific strategy has the added benefit of increasing the accuracies of devices with the worst 10% accuracies and devices with the best 10% accuracies simultaneously. This strategy is built upon the proposed primitive Algorithm 4, and in practice, people can develop other heuristics to improve the performance (similar to what we explore here), based on the method of adaptively averaging model updates proposed in Algorithm 4.

Dataset	**Objective**	**Average** (%)	**Worst 10%** (%)	**Best 10%** (%)	**Variance**	**Angle** (°)	**KL($a\|u$)**
Vehicle	q=0	87.3 ± .5	43.0 ± 1.0	95.7 ± 1.0	291 ± 18	11.3 ± .3	.031 ± .003
	q=5	87.7 ± .7	69.9 ± .6	94.0 ± .9	48 ± 5	4.6 ± .2	.003 ± .000
	multiple q	88.5 ± .3	70.0 ± 2.0	95.8 ± .6	52 ± 7	4.7 ± .3	.004 ± .000
Shakespeare	q=0	51.1 ± .3	39.7 ± 2.8	72.9 ± 6.7	82 ± 41	9.8 ± 2.7	.014 ± .006
	q=.001	52.1 ± .3	42.1 ± 2.1	69.0 ± 4.4	54 ± 27	7.9 ± 2.3	.009 ± .05
	multiple q	52.0 ± 1.5	41.0 ± 4.3	72.0 ± 4.8	72 ± 32	10.1 ± .7	.017 ± .000

Convergence Speed of q-FFL. Since q-FFL ($q > 0$) is more difficult to optimize, a natural question one might ask is: will the q-FFL $q > 0$ objectives slow the convergence compared with FedAvg? We empirically investigate this on the four datasets. We use q-FedAvg to solve q-FFL, and compare it with FedAvg (i.e., solving q-FFL with $q = 0$). As demonstrated in Figure 4.8, the q values that result in more fair solutions also do not significantly slow down convergence.

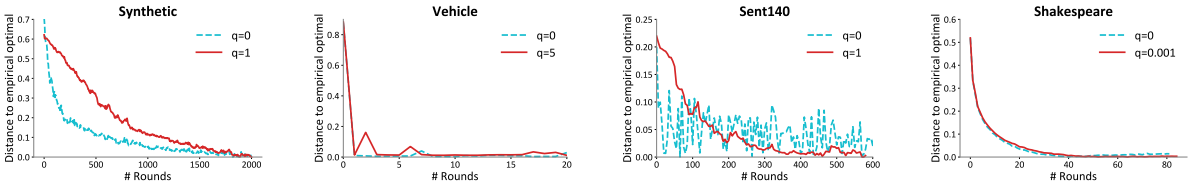


Figure 4.8: The convergence speed of q-FFL compared with FedAvg. We plot the distance to the highest accuracy achieved versus communication rounds. Although q-FFL with q>0 is a more difficult optimization problem, for the q values we choose that could lead to more fair results, the convergence speed is comparable to that of $q = 0$.

Efficiency of q-FFL Compared with AFL. One added benefit of q-FFL is that it leads to faster convergence than AFL—even when we use *non-local-updating* methods for both objectives. In Figure 4.9, we show with respect to the final testing accuracy for the single worst device (i.e., the objective that AFL is trying to optimize), q-FFL converges faster

than AFL. As the number of devices increases (from Fashion MNIST to Vehicle), the performance gap between AFL and q-FFL becomes larger because AFL introduces larger variance.

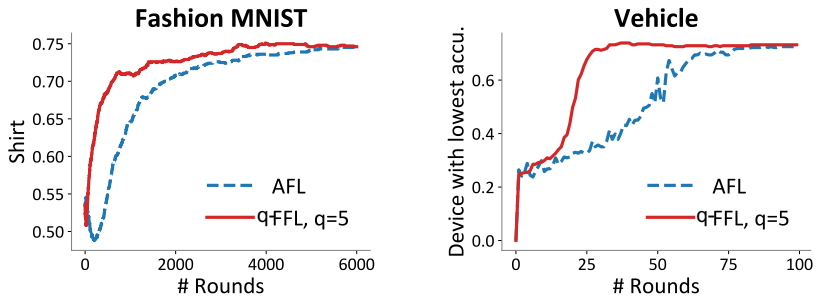


Figure 4.9: q-FFL is more efficient than AFL. With the worst device achieving the same final testing accuracy, q-FFL converges faster than AFL. For Vehicle (with 23 devices) as opposed to Fashion MNIST (with 3 devices), we see that the performance gap is larger. We run full gradient descent at each round for both methods.

Efficiency of q-FedAvg under Different Data Heterogeneity. As mentioned in Appendix 4.7.1, one potential cause for the slower convergence of q-FedAvg on the synthetic dataset may be that local updating schemes could hurt convergence when local data distributions are highly heterogeneous. Although it has been shown that applying updates locally results in significantly faster convergence in terms of communication rounds [209, 266], which is consistent with our observation on most datasets, we note that when data is highly heterogeneous, local updating may hurt convergence. We validate this by creating an IID synthetic dataset (Synthetic-IID) where local data on each device follow the same global distribution. We call the synthetic dataset used in Section 4.3 Synthetic-Non-IID. We also create a hybrid dataset (Synthetic-Hybrid) where half of the total devices are assigned IID data from the same distribution, and half of the total devices are assigned data from different distributions. We observe that if data is perfectly IID, q-FedAvg is more efficient than q-FedSGD. As data become more heterogeneous, q-FedAvg converges more slowly than q-FedSGD in terms of communication rounds. For all three synthetic datasets, we repeat the process of tuning a best constant step-size for FedSGD and observe similar results as before — our dynamic solver q-FedSGD behaves similarly (or even outperforms) a best hand-tuned FedSGD.

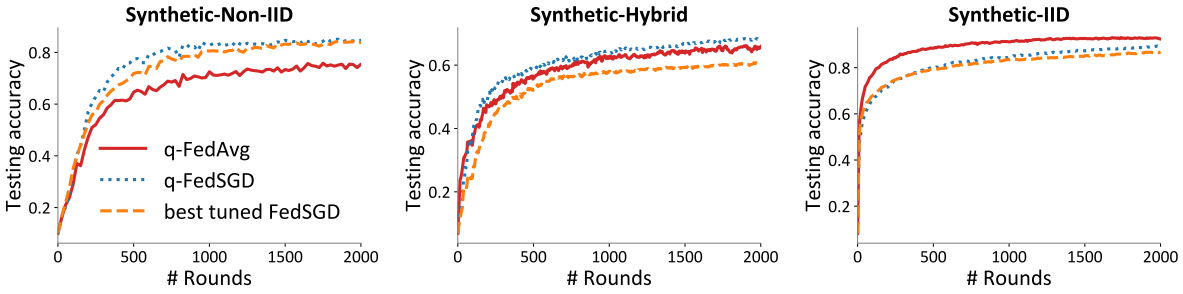


Figure 4.10: Convergence of q-FedAvg compared with q-FedSGD under different data heterogeneity. When data distributions are heterogeneous, it is possible that q-FedAvg converges more slowly than q-FedSGD. Again, the proposed dynamic solver q-FedSGD performs similarly (or better) than a best tuned fixed-step-size FedSGD.

Chapter 5

Robustness: Addressing Competing Constraints Through Personalization

In federated learning, heterogeneity not only affects separate metrics such as accuracy or fairness, but also affects how they interplay with each other. In this chapter, we explore this direction by considering the constraints between fairness and robustness, and propose a unified framework to probably address them simultaneously.

5.1 Overview

To deploy FL in practice, it is necessary for the resulting systems to be not only accurate, but to also satisfy a number of pragmatic constraints regarding issues such as fairness, robustness, and privacy. Simultaneously satisfying these varied constraints can be exceptionally difficult [144].

We focus in this work specifically on issues of accuracy, fairness (i.e., limiting performance disparities across the network [215]), and robustness (against training-time data and model poisoning attacks). Many prior efforts have separately considered fairness or robustness in federated learning. For instance, fairness strategies include using minimax optimization to focus on the worst-performing devices [121, 215] or reweighting the devices to allow for a flexible fairness/accuracy tradeoff [182, 184]. Robust methods commonly use techniques such as gradient clipping [277] or robust aggregation [35, 309].

While these approaches may be effective at either promoting fairness or defending against training-time attacks in isolation, we show that the constraints of fairness and robustness can directly compete with one another when training a single global model, and that simultaneously optimizing for accuracy, fairness, and robustness requires careful consideration. For example, as we empirically demonstrate (Section 5.3), current fairness approaches can render FL systems highly susceptible to training time attacks from malicious devices. On the other hand, robust methods may filter out rare but informative updates, causing unfairness [288].

In this work, we investigate a simple, scalable technique to simultaneously improve accuracy, fairness, and robustness in federated learning. While addressing the competing

constraints of FL may seem like an insurmountable problem, we identify that statistical heterogeneity (i.e., non-identically distributed data) is a root cause for tension between these constraints, and is key in paving a path forward. In particular, we suggest that methods for personalized FL—which model and adapt to the heterogeneity in federated settings by learning distinct models for each device—may provide *inherent* benefits in terms of fairness and robustness.

To explore this idea, we propose `Ditto`, a scalable federated multi-task learning framework. `Ditto` can be seen as a lightweight personalization add-on for standard global FL. It is applicable to both convex and non-convex objectives, and inherits similar privacy and efficiency properties as traditional FL. We evaluate `Ditto` on a suite of federated benchmarks and show that, surprisingly, this simple form of personalization can in fact deliver better accuracy, robustness, and fairness benefits than state-of-the-art, problem-specific objectives that consider these constraints separately. We summarize our contributions below:

- We propose `Ditto`, a multi-task learning objective for federated learning that provides personalization while retaining similar efficiency and privacy benefits as traditional FL. We provide convergence guarantees for our proposed `Ditto` solver, which incorporate common practices in cross-device federated learning such as limited device participation and local updating. Despite its simplicity, we show that `Ditto` can deliver similar or superior accuracy relative to other common methods for personalized federated learning.
- Next, we demonstrate that the benefits of `Ditto` go beyond accuracy—showing that the personalized objective can inherently offer *robustness* superior to that of common robust FL methods across a diverse set of data and model poisoning attacks. On average across all datasets and attacks, `Ditto` improves test accuracy by $\sim$6% (absolute) over the strongest robust baseline.
- Similarly, we show that `Ditto` can naturally increase *fairness*—reducing variance of the test accuracy across devices by $\sim$10% while maintaining similar or superior accuracy relative to state-of-the-art methods for fair FL.
- Finally, we highlight that `Ditto` is particularly useful for practical applications where we simultaneously care about multiple constraints (accuracy, fairness, and robustness). We motivate this through analysis on a toy example in Section 5.2, as well as experiments across a suite of federated datasets in Section 5.3.

5.2 `Ditto`: Global-Regularized Federated Multi-Task Learning

In order to explore the possible fairness/robustness benefits of personalized FL, we first propose a simple and scalable framework for federated multi-task learning. As we will see, this lightweight personalization framework is amenable to analyses while

also having strong empirical performance. We explain our proposed objective, Ditto, in Section 5.2.1 and then present a scalable algorithm to solve it in federated settings (Section 5.2.2). We provide convergence guarantees for our solver, and explain several practical benefits of our modular approach in terms of privacy and efficiency. Finally, in Section 5.2.3, we characterize the benefits of Ditto in terms of fairness and robustness on a class of linear problems. We empirically explore the fairness and robustness properties against state-of-the-art baselines in Section 5.3.

5.2.1 Ditto Objective

Traditionally, federated learning objectives consider fitting a single global model, w, across all local data in the network. The aim is to solve:

$$\min_{w} \; G(F_1(w), \dots \; F_K(w)) \,, \tag{Global Obj}$$

where $F_k(w)$ is the local objective for device k, and $G(\cdot)$ is a function that aggregates the local objectives $\{F_k(w)\}_{k\in[K]}$ from each device. For example, in FedAvg [206], $G(\cdot)$ is typically set to be a weighted average of local losses, i.e., $\sum_{k=1}^{K} p_k F_k(w)$, where p_k is a pre-defined non-negative weight such that $\sum_k p_k = 1$.

However, in general, each device may generate data x_k via a distinct distribution $\mathcal{D}_k$, i.e., $F_k(w) := \mathbb{E}_{x_k \sim \mathcal{D}_k}\left[f_k(w; x_k)\right]$. To better account for this heterogeneity, it is common to consider techniques that learn personalized, device-specific models, $\{v_k\}_{k\in[K]}$ across the network. In this work we explore personalization through a simple framework for federated multi-task learning. We consider two 'tasks': the global objective (Global Obj), and the local objective $F_k(v_k)$, which aims to learn a model using only the data of device k. To relate these tasks, we incorporate a regularization term that encourages the personalized models to be close to the optimal global model. The resulting bi-level optimization problem for each device $k \in [K]$ is given by:

$$\begin{aligned} \min_{v_k} \quad & h_k(v_k; w^*) := F_k(v_k) + \frac{\lambda}{2}\left\|v_k - w^*\right\|^2 \\ \text{s.t.} \quad & w^* \in \arg\min_{w} G(F_1(w), \dots \; F_K(w))) \,. \end{aligned} \tag{Ditto}$$

Here the hyperparameter λ controls the interpolation between local and global models. When λ is set to 0, Ditto is reduced to training local models; as λ grows large, it recovers global model objective (Global Obj) ($\lambda \to +\infty$).

Intuition for Fairness/Robustness Benefits. In addition to improving accuracy via personalization, we argue that Ditto can offer fairness and robustness benefits. To reason about this, consider a simple case where data are *homogeneous* across devices. Without adversaries, learning a single global model is optimal for generalization. However, in the presence of adversaries, learning globally might introduce corruption, while learning local models may not generalize well due to limited sample size. Ditto with an appropriate value of λ offers a tradeoff between these two extremes: the smaller λ,

the more the personalized models v_k can deviate from the (corrupted) global model w, potentially providing robustness at the expense of generalization. In the heterogeneous case (which can lead to issues of unfairness), a finite λ exists to offer robustness and fairness jointly. We explore these ideas more rigorously in Section 5.2.3 by analyzing the tradeoffs between accuracy, fairness, and robustness in terms of λ for a class of linear regression problems, and demonstrate fairness/robustness benefits of `Ditto` empirically in Section 5.3.

Other Personalization Schemes. Personalization is a widely-studied topic in FL. Our intuition in `Ditto` is that personalization, by reducing reliance on the global model, can reduce representation disparity (i.e., unfairness) and potentially improve robustness. It is possible that other personalization techniques beyond `Ditto` offer similar benefits: We provide some initial, encouraging results on this in Section 5.3.4. However, we specifically explore `Ditto` due to its simple nature, scalability, and strong empirical performance. `Ditto` is closely related to works that regularize personalized models towards their average [72, 107, 108], similar to classical mean-regularized MTL [87]; `Ditto` differs by regularizing towards a global model rather than the average personalized model. We find that this provides benefits in terms of *analysis* (Section 5.2.3), as we can easily reason about `Ditto` relative to the global ($\lambda \rightarrow \infty$) vs. local ($\lambda \rightarrow 0$) baselines; *empirically*, in terms of accuracy, fairness, and robustness (Section 5.3); and *practically*, in terms of the modularity it affords our corresponding solver (Section 5.2.2).

Other Regularizers. To encourage the personalized models v_k to be close to the optimal global model w^*, there are choices beyond the L_2 norm that could be considered, e.g., using a Bregman divergence-based regularizer or reshaping the L_2 ball using the Fisher information matrix. Under the logistic loss (used in our experiments), the Bregman divergence will reduce to KL divergence, and its second-order Taylor expansion will result in an L_2 ball reshaped with the Fisher information matrix. Such regularizers are studied in other related contexts like continual learning [156, 251], multi-task learning [312], or fine-tuning for language models [129]. However, in our experiments (Section 5.3.4), we find that incorporating approximate empirical Fisher information [156, 312] or symmetrized KL divergence [129] does not improve the performance over the simple L_2 regularized objective, while adding non-trivial computational overhead.

Remark (Relation to FedProx). We note that the L_2 term in `Ditto` bears resemblance to FedProx, a method which was developed to address heterogeneity in federated optimization [176]. However, `Ditto` fundamentally differs from FedProx in that the goal is to learn *personalized* models v_k, while FedProx produces a single global model w. For instance, when the regularization hyperparameter is zero, `Ditto` reduces to learning separate local models, whereas FedProx would reduce to FedAvg. In fact, `Ditto` is significantly more general than FedProx in that FedProx could be used as the global model solver in `Ditto` to optimize $G(\cdot)$. As discussed above, other regularizers beyond the L_2 norm may also be used in practice.

We note that Hu et al. [121] propose FedMGDA+, a method targeting fair and robust FL; however, this work combines classical fairness (minimax optimization) and robustness (gradient normalization) techniques, in contrast to the multi-task learning framework proposed herein, which we show can *inherently* provide benefits with respect to both constraints simultaneously.

5.2.2 `Ditto` Solver

To solve `Ditto`, we propose jointly solving for the global model w^* and personalized models $\{v_k\}_{k\in[K]}$ in an alternating fashion, as summarized in Algorithm 6. Optimization proceeds in two phases: (i) updates to the global model, w^*, are computed across the network, and then (ii) the personalized models v_k are fit on each local device. The process of optimizing w^* is exactly the same as optimizing for any objective $G(\cdot)$ in federated settings: If we use iterative solvers, then at each communication round, each selected device can solve the local subproblem of $G(\cdot)$ approximately (Line 5). For personalization, device k solves the global-regularized local objective $\min_{v_k} h_k(v_k; w^t)$ inexactly at each round (Line 6-7). Due to this alternating scheme, our solver can scale well to large networks, as it does not introduce additional communication or privacy overheads compared with existing solvers for $G(\cdot)$. In our experiments (all except Table 5.3), we use FedAvg as the objective and solver for $G(\cdot)$, under which we simply let device k run local SGD on F_k (Line 5).

We note that another natural choice to solve the `Ditto` objective is to first obtain w^*, and then for each device k, perform finetuning on the local objective $\min_{v_k} h_k(v_k; w^*)$. These two approaches will arrive at the same solutions in strongly convex cases. In non-convex settings, we observe that there may be additional benefits of joint optimization: Empirically, we find that the updating scheme tends to guide the optimization trajectory towards a better solution compared with finetuning starting from w^*, particularly when w^* is corrupted by adversarial attacks (Section 5.3.4). Intuitively, under training-time attacks, the global model may start from a random one, get optimized, and gradually become corrupted as training proceeds [174]. In these cases, feeding in *early* global information (i.e., before the global model converges to w^*) may be helpful under strong attacks.

We note that `Ditto` with joint optimization requires the devices to maintain local states (i.e., personalized models) and carry these local states to the next communication round where they are selected. Solving `Ditto` with finetuning does not need devices to be stateful, while losing the benefits of alternate updating discussed above.

Modularity of `Ditto`. From the `Ditto` objective and Alg 6, we see that a key advantage of `Ditto` is its modularity, i.e., that we can readily use prior art developed for the Global Obj along with the personalization add-on of $h_k(v_k; w^*)$, as highlighted in red. This has several benefits:

- *Optimization:* It is possible to plug in other methods beyond FedAvg [e.g., 148, 179, 235] in Algorithm 6 to update the global model, and inherit the convergence benefits, if any

Algorithm 6 `Ditto` for Personalized FL

Input: K, T, s, λ, η, w^0, $\{v_k^0\}_{k\in[K]}$
for $t = 0, \cdots, T-1$ **do**
  Server randomly selects a subset of devices S_t, and sends w^t to them
  for device $k \in S_t$ in parallel **do**
    Solve the local sub-problem of $G(\cdot)$ inexactly starting from w^t to obtain w_k^t:

$$w_k^t \leftarrow \text{UPDATE\_GLOBAL}(w^t, \nabla F_k(w^t))$$

    /* Solve $h_k(v_k; w^t)$ */
    Update v_k for s local iterations:

$$v_k = v_k - \eta(\nabla F_k(v_k) + \lambda(v_k - w^t))$$

    Send $\Delta_k^t := w_k^t - w^t$ back
  end for
  Server aggregates $\{\Delta_k^t\}$:

$$w^{t+1} \leftarrow \text{AGGREGATE}\left(w^t, \{\Delta_k^t\}_{k\in\{S_t\}}\right)$$

end for
Return $\{v_k\}_{k\in[K]}$ (personalized), w^T (global)

(we make this more precise in Theorem 5).

- *Privacy:* `Ditto` communicates the same information over the network as typical FL solvers for the global objective, thus preserving whatever privacy or communication benefits exist for the global objective and its respective solver. This is different from most other personalization methods where global model updates depend on local parameters, which may raise privacy concerns [196].
- *Robustness:* Beyond the inherent robustness benefits of personalization, robust global methods can be used with `Ditto` to further improve performance (see Section 5.3.4).

In particular, while not the main focus of our work, we note that `Ditto` may offer a better *privacy-utility* tradeoff than training a global model. For instance, when training `Ditto`, if we fix the number of communication rounds and add the same amount of noise per round to satisfy differential privacy, `Ditto` consumes exactly the same privacy budget as normal global training, while yielding higher accuracy via personalization (Section 5.3). Similar benefits have been studied, e.g., via finetuning strategies [312].

Convergence of Algorithm 6. Note that optimizing the global model w^t does not depend on any personalized models $\{v_k\}_{k\in[K]}$. Therefore, w enjoys the same global convergence rates with the solver we use for G. Under this observation, we present the local convergence of Algorithm 6.

Theorem 5 (Local convergence of Alg. 6; formal statement and proof in Theorem 14)**.** *Assume for $k \in [K]$, F_k is strongly convex and smooth, under common assumptions, if w^t converges to w^* with rate $g(t)$, then there exists a constant $C<\infty$ such that for $\lambda \in \mathbb{R}$, and for $k \in [K]$, v_k^t converges to $v_k^* := \arg\min_{v_k} h_k(v_k; w^*)$ with rate $Cg(t)$.*

Using Theorem 5, we can directly plug in previous convergence analyses for any $G(\cdot)$. For instance, when the global objective and its solver are those of FedAvg, we can obtain an $O(1/t)$ convergence rate for `Ditto` under suitable conditions (Corollary 4). We provide a full theorem statement and proof of convergence in Appendix 5.5.

5.2.3 Analyzing the Fairness/Robustness Benefits of `Ditto` in Simplified Settings

In this section, we more rigorously explore the fairness/robustness benefits of `Ditto` on a class of linear problems. Throughout our analysis, we assume $G(\cdot)$ is the standard objective in FedAvg [206].

Point Estimation. To provide intuition, we first examine a toy one-dimensional point estimation problem. Denote the underlying models for the devices as $\{v_k\}_{k\in[K]}$, $v_k \in \mathbb{R}$, and let the points on device k, $\{x_{k,1}, \ldots, x_{k,n}\}$[1], be observations of v_k with random perturbation, i.e., $x_{k,i} = v_k + z_{k,i}$, where $z_{k,i} \sim \mathcal{N}(0, \sigma^2)$ and are IID. Assume $v_k \sim \mathcal{N}(\theta, \tau^2)$, where θ is drawn from the uniform uninformative prior on $\mathbb{R}$, and τ is a known constant.

Here, τ controls the degree of relatedness of the data on different devices: τ=0 captures the case where the data on all devices are identically distributed while $\tau \to \infty$ results in the scenario where the data on different devices are completely unrelated. The local objective is $\min_{v_k} F_k(v_k) = \frac{1}{2}(v_k - \frac{1}{n_k}\sum_{i=1}^{n_k} x_{k,i})^2$. In the presence of adversaries, we look at a specific type of label poisoning attack. Let K_a denote the number of malicious devices, and the 'capability' of an adversary is modeled by τ_a, i.e., the underlying model of an adversary follows $\mathcal{N}(\theta, \tau_a^2)$ where $\tau_a^2 > \tau^2$.

We first derive the Bayes estimator (which will be the most accurate and robust) for the real model distribution by observing a finite number of training points. Then, we show that by solving `Ditto`, we are able to recover the Bayes estimator with a proper λ^* (with the knowledge of τ). In addition, *the same* λ^* results in the most fair solution among the set of solutions of `Ditto` parameterized by λ. This shows that `Ditto` with a proper choice of λ is Bayes optimal for this particular problem instance. In general, in our theorems, we prove that

$$\lambda^* = \frac{\sigma^2}{n} \frac{K}{K\tau^2 + \frac{K_a}{K-1}(\tau_a^2 - \tau^2)}.$$

We see that λ^* decreases when (i) there are more local samples n, (ii) the devices are less related (larger τ), or (iii) the attacks are stronger (larger number of attackers, K_a,

[1]For ease of notation, we assume each device has the same number of training samples. It is straightforward to extend the current analysis to allow for varying number of samples per device.

and more powerful adversaries, τ_a). Related theorems (Theorem 10-13) are presented in Appendix 5.4.3.

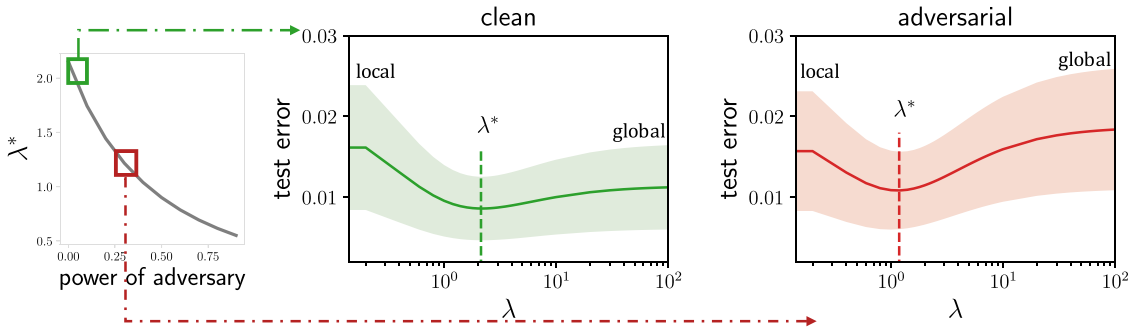


Figure 5.1: Empirically, the λ^* given by our theorems results in the most accurate, fair, and robust solution within Ditto's solution space. λ^* is also optimal in terms of accuracy and robustness among any possible federated estimation algorithms.

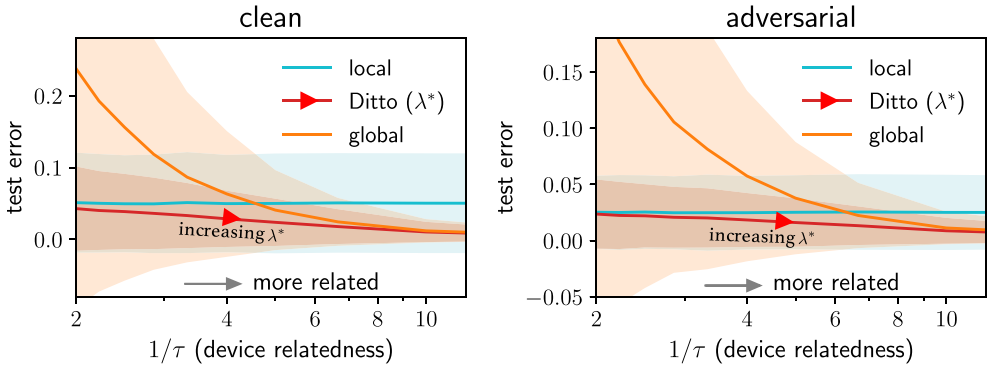


Figure 5.2: Impact of data relatedness across all devices. When $1/\tau$ is small (less related), local outperforms global; when $1/\tau$ is large (more related), global is better than local. Ditto (λ^*) achieves the lowest test error and variance (measured across benign devices).

In Figure 5.1, we plot average test error, fairness (standard deviation shown as error bars), and robustness (test error in the adversarial case) across a set of λ's for both clean and adversarial cases. We see that in the solution space of Ditto, there exists a specific λ which minimizes the average test error and standard deviation across all devices *at the same time*, which is equal to the optimal λ^* given by our theory. Figure 5.2 shows (i) Ditto with λ^* is superior than learning local or global models, and (ii) λ^* should increase as the relatedness between devices ($1/\tau$) increases.

Linear Regression. All results discussed above can be generalized to establish the optimality of Ditto on a class of linear regression problems (with additional assumptions

on feature covariance). We defer readers to Appendix 5.4.2 for full statements and proofs. While our analyses here are limited to a simplified set of attacks and problem settings, we build on this intuition in Section 5.3—empirically demonstrating the accuracy, robustness, and fairness benefits of Ditto using both convex and non-convex models, across a range of federated learning benchmarks, and under a diverse set of attacks.

5.3 Experiments

In this section, we first demonstrate that Ditto can inherently offer similar or superior robustness relative to strong robust baselines (Section 5.3.1). We then show it results more fair performance than recent fair methods (Section 5.3.2). Ditto is particularly well-suited for mitigating the tension between these constraints and achieving both fairness and robustness simultaneously (Section 5.3.3). We explore additional beneficial properties of Ditto in Section 5.3.4.

Setup. For all experiments, we measure robustness via test accuracy, and fairness via test accuracy variance (or standard deviation), both across benign devices (see Def. 2, 1). We use datasets from common FL benchmarks [1, 43, 265], which cover both vision and language tasks, and convex and non-convex models. Detailed datasets and models are provided in Table 5.4 in Appendix 5.6. We split local data on each device into train/test/validation sets randomly, and measure performance on the test data. For each device, we select λ locally based on its local validation data. We further assume the devices can make a binary decision on whether the attack is strong or not. For devices with very few validation samples (less than 4), we use a fixed small λ (λ=0.1) for strong attacks, and use a fixed relatively large λ (λ=1) for all other attacks. For devices with more than 5 validation data points, we let each select λ from $\{0.05, 0.1, 0.2\}$ for strong attacks, and select λ from $\{0.1, 1, 2\}$ for all other attacks. See Appendix 5.7.2 for details. More advanced tuning methods are left for future work. Our code, data, and experiments are publicly available at github.com/litian96/ditto.

5.3.1 Robustness of Ditto

Following our threat model described in Definition 2, we apply three attacks to corrupt a random subset of devices. We pick corruption levels until a point where there is a significant performance drop when training a global model. We compare robustness (Def. 2) of Ditto with various defense baselines, presenting the results of three strongest defenses in Figure 5.3. Execution details and full results are reported in Appendix 5.7.4. As shown in Figure 5.3, Ditto achieves the highest accuracy under most attacks, particularly those with a large fraction of malicious devices. On average across all datasets and attacks, Ditto results in $\sim$6% absolute accuracy improvement compared with the strongest robust baseline (Appendix 5.7.4). In scenarios where a robust baseline outperforms Ditto, we have also found that replacing the global objective and its solver (FedAvg) with a robust

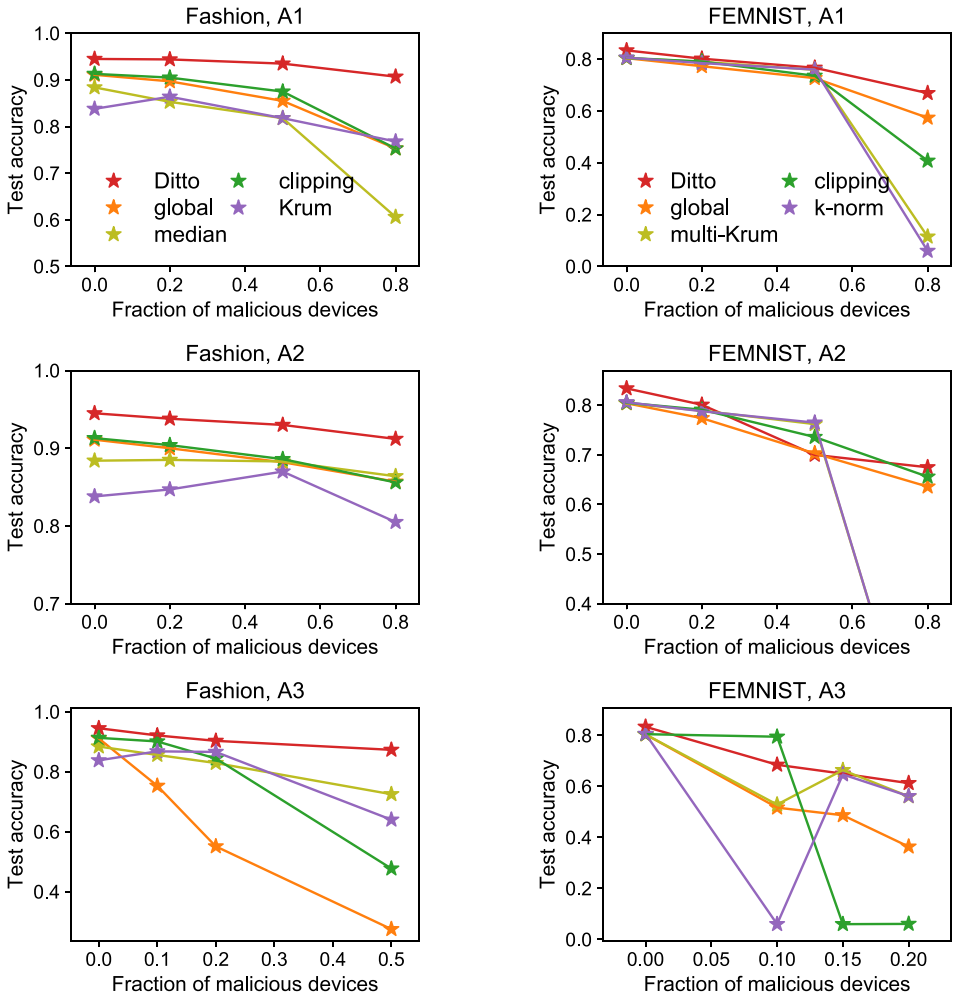


Figure 5.3: Robustness, i.e., average test accuracy on benign devices (Definition 2), on Fashion MNIST and FEMNIST. We compare `Ditto` with learning a global model and three strong defense mechanisms (see Appendix 5.7 for results on all defense baselines), and find that `Ditto` is the most robust under almost all attacks.

version (e.g., using robust aggregators) can further improve `Ditto`, yielding superior performance (Section 5.3.4).

5.3.2 Fairness of `Ditto`

To explore the fairness of `Ditto`, we compare against TERM [184] as a baseline. It is an improved version of the q-FFL [182] objective, which has been recently proposed for fair federated learning. TERM also recovers AFL [215], another fair FL objective, as a special case. TERM uses a parameter t to offer flexible tradeoffs between fairness and accuracy. In Table 5.1, we compare the proposed objective with global, local, and fair methods (TERM) in terms of test accuracies and standard deviation. When the corruption

Table 5.1: **Average (standard deviation)** test accuracy to benchmark performance and fairness (Definition 1) on Fashion MNIST and FEMNIST. `Ditto` is either (i) more fair compared with the baselines of training a global model, or (ii) more accurate than the fair baseline under a set of attacks. We bold the method with highest average minus standard deviation across all methods.

Fashion		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
global	.911 (.08)	.897 (.08)	.855 (.10)	.753 (.13)	.900 (.08)	.882 (.09)	.857 (.10)	.753 (.10)	.551 (.13)	.275 (.12)
local	.876 (.10)	.874 (.10)	.876 (.11)	.879 (.10)	.874 (.10)	.876 (.11)	.879 (.10)	.877 (.10)	.874 (.10)	**.876 (.11)**
fair (TERM, t=1)	.909 (.07)	.751 (.12)	.637 (.13)	.547 (.11)	.731 (.13)	.637 (.14)	.635 (.14)	.653 (.13)	.601 (.12)	.131 (.16)
`Ditto`	**.943 (.06)**	**.944 (.07)**	**.937 (.07)**	**.907 (.10)**	**.938 (.07)**	**.930 (.08)**	**.913 (.09)**	**.921 (.09)**	**.902 (.09)**	.873 (.11)
FEMNIST		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	.804 (.11)	.773 (.11)	.727 (.12)	.574 (.15)	.774 (.11)	**.703 (.14)**	.636 (.15)	.517 (.14)	.487 (.14)	.314 (.13)
local	.628 (.15)	.620 (.14)	.627 (.14)	.607 (.14)	.620 (.14)	.627 (.14)	.607 (.14)	.622 (.14)	.621 (.14)	**.620 (.14)**
fair (TERM, t=1)	.809 (.11)	.636 (.15)	.562 (.13)	.478 (.12)	.440 (.15)	.336 (.12)	.363 (.12)	.353 (.12)	.316 (.12)	.299 (.11)
`Ditto`	**.834 (.09)**	**.802 (.10)**	**.762 (.11)**	**.672 (.13)**	**.801 (.09)**	.700 (.15)	**.675 (.14)**	**.685 (.15)**	**.650 (.14)**	.613 (.13)

level is high, 'global' or 'fair' will even fail to converge. `Ditto` results in more accurate and fair solutions both with and without attacks. On average across all datasets, `Ditto` reduces variance across devices by $\sim$10% while improving absolute test accuracy by 5% compared with TERM (on clean data).

5.3.3 Addressing Competing Constraints

In this section, we examine the competing constraints between robustness and fairness. When training a single global model, fair methods aim to encourage a more uniform performance distribution, but may be highly susceptible to training-time attacks in statistically heterogeneous environments. We investigate the test accuracy on benign devices when learning global, local, and fair models. In the TERM objective, we set $t = 1, 2, 5$ to achieve different levels of fairness (the higher, the fairer). We perform the data poisoning attack (A1 in Def. 2). The results are plotted in Figure 5.4. As the corruption level increases, we see that fitting a global model becomes less robust. Using fair methods will be more susceptible to attacks. When t gets larger, the test accuracy gets lower, an indication that the fair method is overfitting to the corrupted devices relative to the global baseline.

Next, we apply various strong robust methods under the same attack, and explore the robustness/accuracy and fairness performance. The robust approaches include: Krum, multi-Krum [35], taking the coordinate-wise median of gradients ('median'), gradient clipping ('clipping'), filtering out the gradients with largest norms ('k-norm'), and taking the gradient of the k-th largest loss where k is the number of malicious devices ('k-loss'). For Krum, multi-Krum, k-norm, and k-loss, we assume that the server knows the expected number of malicious devices that are selected each round, and can set k

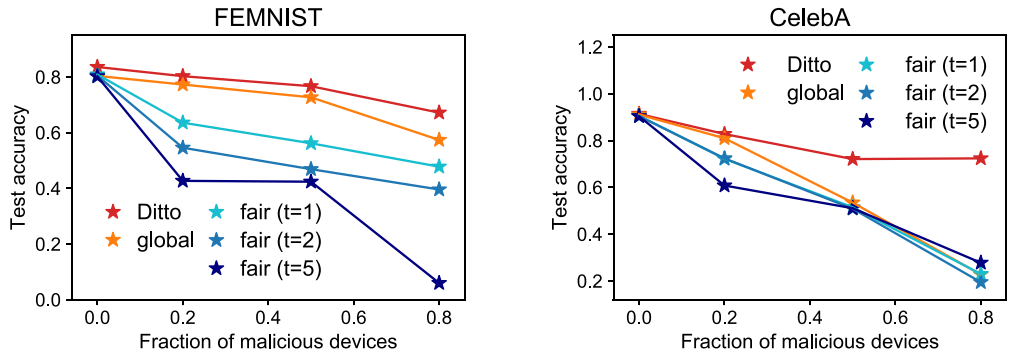


Figure 5.4: Fair methods can overfit to corrupted devices (possibly with large training losses) by imposing more weights on them, thus being particularly susceptible to attacks.

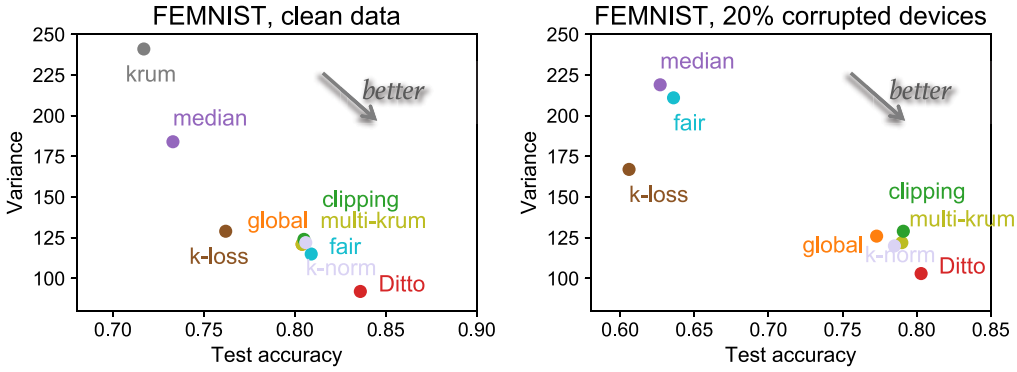


Figure 5.5: Compared with learning a global model, robust baselines (i.e., the methods listed in the figure excluding 'global' and 'Ditto') are either robust but not fair (with higher accuracy, larger variance), or not even robust (with lower accuracy). `Ditto` lies at the lower right corner, which is our preferred region.

accordingly for k-norm and k-loss. From Figure 5.5, we see that robust baselines are either (i) more robust than global but less fair, or (ii) fail to provide robustness due to heterogeneity. `Ditto` is more robust, accurate, and fair.

5.3.4 Additional Properties of `Ditto`

Personalization. We additionally explore the performance of other personalized FL methods in terms of accuracy and fairness, on both clean and adversarial cases. In particular, we consider objectives that (i) regularize with the average (L2SGD [107]) or the learnt device relationship matrix (MOCHA [265]), (ii) encourage closeness to the global model in terms of some specific function behavior (EWC [156, 312] and Symmetrized KL (SKL)), (iii) interpolate between local and global models (APFL [69] and mapper [202]), and (iv) have been motivated by meta-learning (Per-FedAvg (HF) [88]). We provide a

detailed description in Appendix 5.6.

We compare Ditto with the above alternatives, using the same learning rate tuned on FedAvg on clean data for all methods except Per-FedAvg, which requires additional tuning to prevent divergence. For finetuning methods (EWC and SKL), we finetune on each local device for 50 epochs starting from the converged global model. We report results of baseline methods using their best hyperparameters. Despite Ditto's simplicity, in Table 5.2 below, we see that Ditto achieves similar or superier test accuracy with slightly lower standard deviation compared with these recent personalization methods.

We also evaluate the performance of MOCHA with a convex SVM model. MOCHA is more robust and fair than most baselines, which is in line with our reasoning that personalization can provide benefits for these constraints. Further understanding the robustness/fairness benefits of other personalized approaches would be an interesting direction of future work.

Table 5.2: Ditto is competitive with or outperforms other recent personalization methods. We report the average (standard deviation) of test accuracies across all devices to capture performance and fairness (Definition 1), respectively.

	Clean		50% Adversaries (A1)	
Methods	**FEMNIST**	**CelebA**	**FEMNIST**	**CelebA**
global	.804 (.11)	.911 (.19)	.727 (.12)	.538 (.28)
local	.628 (.15)	.692 (.27)	.627 (.14)	.682 (.27)
plain finetuning	.815 (.09)	.912 (.18)	.734 (.12)	.721 (.28)
L2SGD	.817 (.10)	.899 (.18)	.732 (.15)	.725 (.25)
EWC	.810 (.11)	.910 (.18)	.756 (.12)	.642 (.26)
SKL	.820 (.10)	**.915 (.16)**	.752 (.12)	.708 (.27)
Per-FedAvg (HF)	.827 (.09)	.907 (.17)	.604 (.14)	**.756 (.26)**
mapper	.792 (.12)	.773 (.25)	.726 (.13)	.704 (.27)
APFL	.811 (.11)	.911 (.17)	.750 (.11)	.710 (.27)
Ditto	**.836 (.10)**	.914 (.18)	**.767 (.10)**	.721 (.27)

Augmenting with Robust Baselines. Ditto allows the flexibility of learning robust w^* leveraging any previous robust aggregation techniques, which could further improve the performance of personalized models. For instance, in the aggregation step at the server side (Line 7 in Algorithm 6), instead of simply averaging the global model updates as in FedAvg, we can aggregate them via multi-Krum, or after gradient clipping. As is shown in Table 5.3, Ditto combined with clipping yields improvements compared with vanilla Ditto.

Comparing Two Solvers. As mentioned in Section 5.2.2, another way to solve Ditto is to finetune on $\min_{v_k} h_k(v_k; w^*)$ for each $k \in [K]$ after obtaining w^*. We examine the performance of two solvers under the model replacement attack (A3) with 20% adversaries. In realistic federated networks, it may be challenging to determine how

Table 5.3: Augmenting `Ditto` with robust baselines can further improve performance.

FEMNIST	**A1**		**A2**		**A3**	
Methods	20%	80%	20%	80%	10%	20%
global	.773	.574	.774	.636	.517	.364
clipping	.791	.408	.791	.656	.795	.061
`Ditto`	.803	**.669**	.792	.681	.695	.650
`Ditto` + clipping	**.810**	.645	**.808**	**.684**	**.813**	**.672**

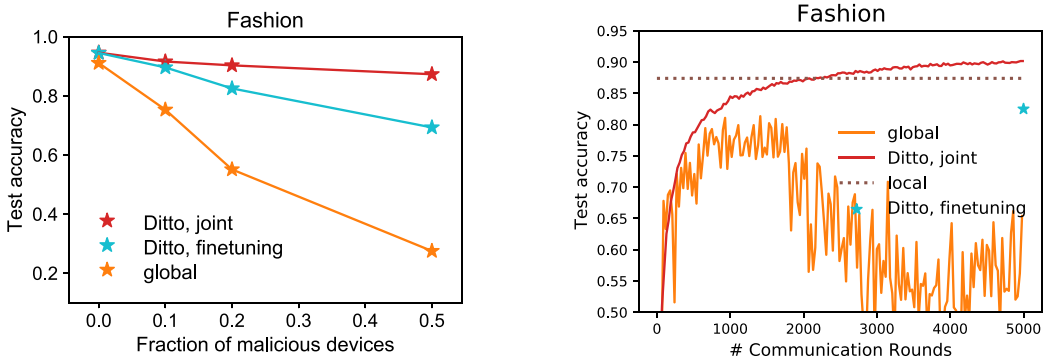


Figure 5.6: `Ditto` with joint optimization (Algorithm 6) outperforms the alternative local finetuning solver under the strong model replacement attack.

many iterations to finetune for, particularly over a heterogeneous network of devices. To obtain the best performance of finetuning, we solve $\min_{v_k} h_k(v_k; w^*)$ on each device by running different iterations of mini-batch SGD and pick the best one. As shown in Figure 5.6, the finetuning solver improves the performance compared with learning a global model, while `Ditto` combined with joint optimization performs the best. One can also perform finetuning after early stopping; however, it is essentially solving a different objective and it is difficult to determine the stopping criteria. We discuss this in more detail in Appendix 5.7.1.

5.4 Analysis of the Federated Multi-Task Learning Objective `Ditto`

Here, we provide theoretical analyses of `Ditto`, mainly on a class of linear models. In this linear setting, we investigate accuracy, fairness, and robustness of `Ditto`. We first discuss some general properties of `Ditto` for strongly convex functions in terms of the training performance in Section 5.4.1. We next present our main results on characterizing the benefits (accuracy, fairness, and robustness) of `Ditto` on linear regression in Section 5.4.2. Finally, we present results on a special case of linear regression (federated point estimation problem examined in Section 5.2.3) in Section 5.4.3.

5.4.1 Properties of `Ditto` for Strongly Convex Functions

Let the `Ditto` objective on device k be

$$h_k(w) = F_k(w) + \lambda\psi(w), \tag{5.1}$$

where F_k is strongly convex, and

$$\psi(w) := \frac{1}{2}\|w - w^*\|^2, \tag{5.2}$$

$$w^* := \arg\min_w \left\{ \frac{1}{K} \sum_{k\in[K]} F_k(w) \right\}. \tag{5.3}$$

Let

$$\widehat{w}_k(\lambda) = \arg\min_w h_k(w). \tag{5.4}$$

Without any distributional assumptions on the tasks, we first characterize the solutions of the objective $h_k(w)$.

Lemma 8. *For all $\lambda \geqslant 0$,*

$$\frac{\partial}{\partial\lambda} F_k(\widehat{w}_k(\lambda)) \geqslant 0, \tag{5.5}$$

$$\frac{\partial}{\partial\lambda} \psi(\widehat{w}_k(\lambda)) \leqslant 0. \tag{5.6}$$

In addition, for all k, if $F_k(w^*)$ is finite, then

$$\lim_{\lambda\to\infty} \widehat{w}_k(\lambda) = w^*. \tag{5.7}$$

Proof. The proof here directly follows the proof in Hanzely and Richtárik [Theorem 3.1, 107]. □

As λ increases, the local empirical training loss $F_k(\widehat{w}_k(\lambda))$ will also increase, and the resulting personalized models will be closer to the global model. Therefore, λ effectively controls how much personalization we impose. Since for any device $k \in [K]$, training loss is minimized when $\lambda = 0$, training separate local models is the most robust and fair *in terms of training performance when we do not consider generalization*.

However, in order to obtain the guarantees on the test performance, we need to explicitly model the joint distribution of data on all devices. In the next section, we explore a Bayesian framework on a class of linear problems to examine the generalization, fairness, and robustness of the `Ditto` objective, all on the underlying test data.

5.4.2 Federated Linear Regression

We first examine the case without corrupted devices in Section 5.4.2.1. We prove that there exists a λ that results in an optimal average test performance among all possible federated learning algorithms, which coincides with the optimal λ in `Ditto`'s solution space in terms of fairness. When there are adversaries, we analyze the robustness benefits of `Ditto` in Section 5.4.2.2. In particular, we show there exists a λ which leads to the highest test accuracy across benign devices (i.e., the most robust) and minimizes the variance of the test error across benign devices (i.e., the most fair) jointly.

Before we proceed, we first state a technical lemma that will be used throughout the analyses.

Lemma 9. *Let θ be drawn from the non-informative uniform prior on $\mathbb{R}^d$. Further, let $\{\phi_k\}_{k\in[K]}$ denote noisy observations of θ with additive zero-mean independent Gaussian noises with covariance matrices $\{\Sigma_k\}_{k\in[K]}$. Let*

$$\Sigma_\theta := \left(\sum_{k\in[K]} \Sigma_k^{-1}\right)^{-1}. \tag{5.8}$$

Then, conditioned on $\{\phi_k\}_{k\in[K]}$, we can write θ as

$$\theta = \Sigma_\theta \sum_{k\in[K]} \Sigma_k^{-1}\phi_k + z,$$

where z is $\mathcal{N}(0, \Sigma_\theta)$ which is independent of $\{\phi_k\}_{k\in[K]}$.

Lemma 9 is a generalization of Lemma 11 presented in Mahdavifar et al. [200] (restated in Lemma 10 below) to the multivariate Gaussian case. The proof also follows from the proof in Mahdavifar et al. [200].

Lemma 10 (Lemma 11 in Mahdavifar et al. [200])**.** *Let θ be drawn from the non-informative uniform prior on $\mathbb{R}$. Further, let $\{\phi_k\}_{k\in[K]}$ denote noisy observations of θ with additive zero-mean independent Gaussian noises with variances $\{\sigma_k^2\}_{k\in[K]}$. Let*

$$\frac{1}{\sigma_\theta^2} := \sum_{k\in[K]} \frac{1}{\sigma_k^2}. \tag{5.9}$$

Then, conditioned on $\{\phi_k\}_{k\in[K]}$, we can write θ as

$$\theta = \sigma_\theta^2 \sum_{k\in[K]} \frac{\phi_k}{\sigma_k^2} + z,$$

where z is $\mathcal{N}(0, \sigma_\theta^2)$ which is independent of $\{\phi_k\}_{k\in[K]}$.

5.4.2.1 No Adversaries: `Ditto` for Accuracy and Fairness

We consider a Bayesian framework. Let θ be drawn from the non-informative prior on $\mathbb{R}^d$, i.e., uniformly distributed on $\mathbb{R}^d$. We assume that K devices have their data distributed with parameters $\{w_k\}_{k\in[K]}$:

$$w_k = \theta + \zeta_k, \tag{5.10}$$

where $\zeta_k \sim \mathcal{N}(0, \tau^2 \mathbf{I}_d)$ are I.I.D, and $\mathbf{I}_d$ denotes the $d \times d$ identity matrix. τ controls the degree of dependence between the tasks on different devices. If $\tau = 0$, then the data on all devices is distributed according to parameter θ, i.e., the tasks are the same, and if $\tau \to \infty$, the tasks on different devices become completely unrelated.

We first derive optimal estimators $\{w_k\}_{k\in[K]}$ for each device w_k given observations $\{X_i, y_i\}_{i\in[K]}$.

Lemma 11. *Assume that we have*

$$y = Xw + z \tag{5.11}$$

where $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times d}$, and $w \in \mathbb{R}^d$, and $z \in \mathbb{R}^n$. Further assume that $z \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$ and w follows the non-informative uniform prior on $\mathbb{R}^d$. Let

$$\widehat{w} = (X^T X)^{-1} X^T y. \tag{5.12}$$

Then, we have $\widehat{w}$ follows a multi-variate normal distribution as follows:

$$\widehat{w} \sim \mathcal{N}\left((X^T X)^{-1} X^T y, \sigma^2 (X^T X)^{-1}\right). \tag{5.13}$$

Lemma 12. *Let*

$$\widehat{w}_i := (X_i^T X_i)^{-1} X_i^T y_i. \tag{5.14}$$

Let

$$\Sigma_i := \sigma^2 (X_i^T X_i)^{-1} + \tau^2 \mathbf{I}_d. \tag{5.15}$$

Further, let

$$\Sigma_\theta^{\setminus k} := \left(\sum_{i\in[K], i\neq k} \Sigma_i^{-1} \right)^{-1}. \tag{5.16}$$

Further let

$$\mu_\theta^{\setminus k} := \Sigma_\theta^{\setminus k} \sum_{i\in[K], i\neq k} \Sigma_i^{-1} \widehat{w}_i \tag{5.17}$$

Then, conditioned on $\{X_i, y_i\}_{i\in[K],i\neq k}$, we can write θ as

$$\theta = \mu_\theta^{\setminus k} + \eta,$$

where η is $\mathcal{N}(0, \Sigma_\theta^{\setminus k})$ which is independent of $\{X_i, y_i\}_{i\in[K],i\neq k}$.

Proof. From Lemma 11, we know $\widehat{w}_i$ is a noisy observation of the underlying w_i with additive covariance $\sigma^2(X_i^T X_i)^{-1}$. For $\{w_k\}_{k\in[K]}$ defined in our setup, $\widehat{w}_i$ is a noisy observation of θ with additive zero mean and covariance $\Sigma_i := \tau^2\mathbf{I}_d + \sigma^2(X_i^T X_i)^{-1}$. The proof completes by applying Lemma 9 to $\{\widehat{w}_i\}_{i\in[K],i\neq k}$. □

Lemma 13. *Let*

$$\Sigma_{w_k}^{\setminus k} := \Sigma_\theta^{\setminus k} + \tau^2\mathbf{I}_d. \tag{5.18}$$

Further, let

$$\Sigma_{w_k} := \left((\Sigma_{w_k}^{\setminus k})^{-1} + (\Sigma_k - \tau^2\mathbf{I}_d)^{-1}\right)^{-1}. \tag{5.19}$$

Conditioned on $\{X_i, y_i\}_{i\in[K]}$, we have

$$w_k = \Sigma_{w_k}(\Sigma_k - \tau^2\mathbf{I}_d)^{-1}\widehat{w}_k + \Sigma_{w_k}(\Sigma_{w_k}^{\setminus k})^{-1}\mu_\theta^{\setminus k} + \zeta_k, \tag{5.20}$$

where $\zeta_k \sim \mathcal{N}(0, \Sigma_{w_k})$.

Proof. $\widehat{w}_k$ is a noisy observation of w_k with additive noise with zero mean and covariance $\sigma^2(X_k^T X_k)^{-1}$ (which is $\Sigma_k - \tau^2\mathbf{I}_d$). From Lemma 12, we know conditioned on $\{X_i, y_i\}_{i\in[K],i\neq k}$, $\mu_\theta^{\setminus k}$ is a noisy observation of θ with covariance $\Sigma_\theta^{\setminus k}$. Hence, with respect to w_k, the covariance is $\Sigma_\theta^{\setminus k} + \tau^2\mathbf{I}_d := \Sigma_{w_k}^{\setminus k}$. The conclusion follows by applying Lemma 9 to $\widehat{w}_k$ and $\mu_\theta^{\setminus k}$. □

Let the empirical loss function of the linear regression problem on device k be

$$F_k(w) = \frac{1}{n}\left\|X_k w - y_k\right\|^2. \tag{5.21}$$

Then the estimator $\widehat{w}_k$ is $(X_k^T X_k)^{-1}X^T y_k$. Applying the previous lemmas, we obtain an optimal estimator w_k given all training samples from K devices (see (5.20)). w_k is Bayes optimal among all solutions that can be achieved by any learning method. Next, we examine the `Ditto` objective and its solution space parameterized by λ.

Let each device solve the following objective

$$\min_w h_k(w) = F_k(w) + \frac{\lambda}{2}\left\|w - w^*\right\|^2, \text{ s.t.} \quad w^* = \frac{1}{K}\arg\min_w \sum_{k=1}^{K} F_k(w). \tag{5.22}$$

The local empirical risk minimizer for each device k is

$$\widehat{w}_k(\lambda) = \left(\frac{1}{n}X_k^\top X_k + \lambda I\right)^{-1}\left(\frac{1}{n}X_k^\top Y_k + \lambda w^*\right) \tag{5.23}$$

$$= \left(\frac{1}{n}X_k^\top X_k + \lambda I\right)^{-1}\left(\left(\frac{1}{n}X_k^\top X_k\right)\widehat{w}_k + \lambda\sum_{k=1}^{K}(X^\top X)^{-1}X_k^\top X_k\widehat{w}_k\right) \tag{5.24}$$

We next prove that for any $k \in [K]$, $\widehat{w}_k(\lambda)$ with a specific λ can achieve the optimal w_k.

Theorem 6. *Assume for any $1 \leqslant i \leqslant K$, $X_i^T X_i = \beta \mathbf{I}_d$ for some constant β. Let λ^* be the optimal λ that minimizes the test performance on device k, i.e.,*

$$\lambda^* = \arg\min_{\lambda} E\left\{ F_k(\widehat{w}_k(\lambda))|\, \widehat{w}_k, \mu_\theta^{\backslash k}\right\}. \tag{5.25}$$

Then,

$$\lambda^* = \frac{\sigma^2}{n\tau^2}. \tag{5.26}$$

Proof. Notice that

$$\arg\min_{\lambda} E\left\{F_k(\widehat{w}_k(\lambda))|\widehat{w}_k, \mu_\theta^{\backslash k}\right\} = \arg\min_{\lambda} E\left\{\|X_k\widehat{w}_k(\lambda) - (X_k w_k + z_k)\|^2|\widehat{w}_k, \mu_\theta^{\backslash k}\right\} \tag{5.27}$$

$$= \arg\min_{\lambda} E\left\{\|X_k\left(\widehat{w}_k(\lambda) - w_k\right)\|^2|\widehat{w}_k, \mu_\theta^{\backslash k}\right\} \tag{5.28}$$

$$= \arg\min_{\lambda} E\left\{\|w_k - \widehat{w}_k(\lambda)\|^2\,|\widehat{w}_k, \mu_\theta^{\backslash k}\right\}. \tag{5.29}$$

Plug in $X_k^T X_k = \beta\mathbf{I}$ into (5.20) and (5.24) respectively, we have the optimal estimator w_k is

$$w_k = \left(\frac{K-1}{\frac{\sigma^2}{\beta} + K\tau^2} + \frac{\beta}{\sigma^2}\right)^{-1}\frac{\beta}{\sigma^2}\widehat{w}_k + \left(\frac{K-1}{\frac{\sigma^2}{\beta} + K\tau^2} + \frac{\beta}{\sigma^2}\right)^{-1}\frac{\beta}{\sigma^2 + K\tau^2\beta}\sum_{i\in[K], i\neq k}\widehat{w}_i + \zeta_k, \tag{5.30}$$

and $\widehat{w}_k(\lambda)$ is

$$\widehat{w}_k(\lambda) = \left(\frac{n}{\beta + n\lambda}\right)\left(\left(\frac{\beta}{n} + \frac{\lambda}{K}\right)\widehat{w}_k + \frac{\lambda}{K}\sum_{i\in[K], i\neq k}\widehat{w}_i\right). \tag{5.31}$$

Taking w_k and $\widehat{w}_k(\lambda)$ into

$$\lambda^* = \arg\min_{\lambda} E\left\{\|w_k - \widehat{w}_k(\lambda)\|_2^2\,|\mu_\theta^{\backslash k}, \widehat{w}_k\right\} \tag{5.32}$$

gives $\lambda^* = \frac{\sigma^2}{n\tau^2}$, as $\widehat{w}_k(\lambda^*)$ is the MMSE estimator of w_k given the observations. □

Remark 4. *We note that by using λ^* in `Ditto`, we not only achieve the most accurate solution for the objective, but also we achieve the most accurate solution of any possible federated linear regression algorithm in this problem, as `Ditto` with λ^* realizes the MMSE estimator for w_k.*

We have derived an optimal $\lambda^* = \frac{\sigma^2}{n\tau^2}$ for `Ditto` in terms of generalization. Recall that we define fairness as the variance of the performance across all devices [113, 182]. Next, we prove that the same λ^* that minimizes the expected MSE also achieves the optimal fairness among all `Ditto` solutions.

Theorem 7. *Assume for any $1 \leqslant i \leqslant K$, $X_i^T X_i = \beta \mathbf{I}_d$ for some constant β. Among all possible solutions `Ditto` parameterized by λ, λ^* results in the most fair performance across all devices when there are no adversaries, i.e., it minimizes the variance of test performance (test loss) across all devices.*

Proof. Denote the variance of test loss across K devices as $\mathbf{Var}_K \left\{ \|X_k \widehat{w}_k(\lambda) - y_k\|_2^2 \right\}$. Let

$$\widehat{E}_k\{a_k\} := \frac{1}{K} \sum_{k \in [K]} a_k. \tag{5.33}$$

Then

$$\begin{aligned}
&\arg\min_{\lambda} \mathbf{Var}_K \left\{ \|X_k \widehat{w}_k(\lambda) - y_k\|_2^2 \right\} \\
&= \arg\min_{\lambda} \mathbf{Var}_K \left\{ \|X_k \widehat{w}_k(\lambda) - (X_k w_k + z_k)\|_2^2 \right\} && (5.34) \\
&= \arg\min_{\lambda} \mathbf{Var}_K \left\{ \|X_k (\widehat{w}_k(\lambda) - w_k)\|_2^2 \right\} && (5.35) \\
&= \arg\min_{\lambda} \mathbf{Var}_K \left\{ \|\widehat{w}_k(\lambda) - w_k\|_2^2 \right\} && (5.36) \\
&= \arg\min_{\lambda} \widehat{E}_K \left\{ \left(\|w_k - \widehat{w}_k\|_2^2 \right)^2 \right\} - \left(\widehat{E}_K \left\{ \|w_k - \widehat{w}_k(\lambda)\|_2^2 \right\} \right)^2. && (5.37)
\end{aligned}$$

Note that

$$w_k - \widehat{w}_k(\lambda) = \zeta + a_k, \tag{5.38}$$

where

$$a_k = \widehat{w}_k(\lambda^*) - \widehat{w}_k(\lambda), \tag{5.39}$$

and $\lambda^* = \frac{\sigma^2}{n\tau^2}$.

We have

$$\begin{aligned}
&\widehat{E}_K \left\{ \left(\|w_k - \widehat{w}_k\|_2^2 \right)^2 \right\} - \left(\widehat{E}_K \left\{ \|w_k - \widehat{w}_k(\lambda)\|_2^2 \right\} \right)^2 && (5.40) \\
&= \widehat{E}_K \left\{ \left(\sum_i^d (w_{ki} - \widehat{w}_k(\lambda)_i)^2 \right)^2 \right\} - \left(\widehat{E}_K \left\{ \sum_i^d (w_{ki} - \widehat{w}_k(\lambda)_i)^2 \right\} \right)^2 && (5.41) \\
&= \widehat{E}_K \left\{ \left(\sum_i^d (\zeta_i + a_{ki})^2 \right)^2 \right\} - \left(\widehat{E}_K \left\{ \sum_i^d (\zeta_i + a_{ki})^2 \right\} \right)^2, && (5.42)
\end{aligned}$$

where w_{ki}, $\widehat{w}_k(\lambda)_i$, ζ_i, and a_{ki} denotes the i-th dimension of w_k, $\widehat{w}_k(\lambda)$, ζ, and a_k and d is the model dimension.

We next expand the variance by decomposing it into two parts. We note

$$\widehat{E}_K\left\{\left(\sum_i^d(\zeta_i+a_{ki})^2\right)^2\right\}-\left(\widehat{E}_K\left\{\sum_i^d(\zeta_i+a_{ki})^2\right\}\right)^2 \tag{5.43}$$

$$=\sum_i^d\widehat{E}_k\left\{(\zeta_i+a_{ki})^4\right\}-\sum_i^d\left(\widehat{E}_K\left\{(\zeta_i+a_{ki})^2\right\}\right)^2 \tag{5.44}$$

$$+2\sum_{i,j\in[d],i\neq j}\widehat{E}_K\left\{(\zeta_i+a_{ki})^2\left(\zeta_j+a_{kj}\right)^2\right\}$$

$$-2\sum_{i,j\in[d],i\neq j}\widehat{E}_K\left\{(\zeta_i+a_{ki})^2\right\}\widehat{E}_K\left\{\left(\zeta_j+a_{kj}\right)^2\right\}. \tag{5.45}$$

For any $i\in[d]$, we have

$$E\left\{\widehat{E}_K\left\{(\zeta_i+a_{ki})^4\right\}-\left(\widehat{E}_K\left\{(\zeta_i+a_{ki})^2\right\}\right)^2\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\} \tag{5.46}$$

$$=E\left\{\widehat{E}_K\left\{\zeta_i^4+6\zeta_i^2a_{ki}^2+a_{ki}^4\right\}-\left(\widehat{E}_K\left\{\zeta_i^2+a_{ki}^2\right\}\right)^2\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\} \tag{5.47}$$

$$=E\left\{\widehat{E}_K\left\{\zeta_i^4+6\zeta_i^2a_{ki}^2+a_{ki}^4\right\}-\left(\widehat{E}_K\left\{\zeta_i^2\right\}\right)^2-2\widehat{E}_K\left\{\zeta_i^2\right\}\widehat{E}_K\left\{a_{ki}^2\right\}-\left(\widehat{E}_K\left\{a_{ki}^2\right\}\right)^2\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\}$$

$$=3\sigma_w^4+6\sigma_w^2\widehat{E}_K\left\{a_{ki}^2\right\}+\widehat{E}_K\left\{a_{ki}^4\right\}-\sigma_w^4-2\sigma_w^2\widehat{E}_K\left\{a_{ki}^2\right\}-\left(\widehat{E}_K\left\{a_{ki}^2\right\}\right)^2 \tag{5.48}$$

$$=2\sigma_w^4+4\sigma_w^2\widehat{E}_K\left\{a_{ki}^2\right\}+\widehat{E}_K\left\{a_{ki}^4\right\}-\left(\widehat{E}_K\left\{a_{ki}^2\right\}\right)^2, \tag{5.49}$$

where σ_w is the i-th diagonal of Σ_{w_k} which is the same across all k's and all dimensions, and we have used the fact that we can swap expectations, and $E\{\zeta_i^4\}=3\sigma_w^4$, given that ζ_i is Gaussian distributed and Σ_{w_k} is a diagonal matrix. For any $i,j\in[d], i\neq j$, we have

$$E\left\{\widehat{E}_K\left(\zeta_i+a_{ki}\right)^2\left(\zeta_j+a_{kj}\right)^2\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\}-E\left\{\widehat{E}_K\left(\zeta_i+a_{ki}\right)^2\widehat{E}_K\left(\zeta_j+a_{kj}\right)^2\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\} \tag{5.50}$$

$$=\widehat{E}_k\{a_{ki}^2a_{kj}^2\}-\widehat{E}_k\{a_{ki}^2\}\widehat{E}_k\{a_{kj}^2\}, \tag{5.51}$$

where we have used the fact that Σ_{w_k} is a diagonal matrix.

Plugging (5.49) and (5.51) into (5.44) and (5.45) yields

$$E\left\{\mathbf{Var}_K\left\{\|\widehat{w}_k(\lambda)-w_k\|_2^2\right\}\middle|\mu_\theta^{\setminus k},\widehat{w}_k\right\} \tag{5.52}$$

$$=2d\sigma_w^4+\sum_i4\sigma_w^2\widehat{E}_k\{a_{ki}^2\}+\sum_i\widehat{E}_k\{a_{ki}^4\}-\sum_i\left(\widehat{E}_k\{a_{ki}^2\}\right)^2+2\sum_{i\neq j}\left(\widehat{E}_k\{a_{ki}^2a_{kj}^2\}-\widehat{E}_k\{a_{ki}^2\}\widehat{E}_k\{a_{kj}^2\}\right)$$

$$= 2d\sigma_w^4 + \sum_i 4\sigma_w^2 \widehat{E}_k\{a_{ki}^2\} + \sum_i \widehat{E}_k\{a_{ki}^4\} + 2\sum_{i\neq j} \widehat{E}_k\{a_{ki}^2 a_{kj}^2\} - (\sum_i \left(\mathbb{E}_k\{a_{ki}^2\}\right)^2 + 2\sum_{i\neq j} \widehat{E}_k\{a_{ki}^2\}\widehat{E}_k\{a_{kj}^2\})\})$$

$$= 2d\sigma_w^4 + \sum_i 4\sigma_w^2 \widehat{E}_k\{a_{ki}^2\} + \widehat{E}_k\{(\sum_i a_{ki}^2)^2\} - (\sum_i \widehat{E}_k\{a_{ki}^2\})^2 \tag{5.53}$$

$$= 2d\sigma_w^4 + \sum_i 4\sigma_w^2 \widehat{E}_k\{a_{ki}^2\} + \frac{1}{K}\sum_k (\sum_i a_{ki}^2)^2 - (\frac{1}{K}\sum_k \sum_i a_{ki}^2)^2 \geqslant 2d\sigma_w^2, \tag{5.54}$$

where setting $\{a_{ki}\}_{1\leqslant k\leqslant K, 1\leqslant i\leqslant d} = 0$ achieves the minimum. □

Observations. From the optimal $\lambda^* = \frac{\sigma^2}{n\tau^2}$ for mean test accuracy and variance of the test accuracy, we have the following observations.

- Test error and variance can be jointly minimized with one λ.
- As $n \to \infty$, $\lambda^* \to 0$, i.e., when each local device has an infinite number of samples, there is no need for federated learning, and training local models is optimal in terms of generalization and fairness.
- As $\tau \to \infty$, $\lambda^* \to 0$, i.e., if the data on different devices (the tasks) are unrelated, then training local models is optimal; On the other hand, as $\tau \to 0$, $\lambda^* \to \infty$, i.e., if the data across all devices are identically distributed, or equivalently if the tasks are the same, then training a global model is the best we can achieve.

So far we have proved that the same λ^* achieves the best performance (expected mean square error) for any device *k and* fairness (variance of mean square error) without considering adversaries. In Section 5.4.2.2 below, we analyze the benefits of `Ditto` for fairness and robustness in the presence of adversaries.

5.4.2.2 With Adversaries: `Ditto` for Accuracy, Fairness, and Robustness

As a special case of data poisoning attacks defined in our threat model (Definition 2), we make the following assumptions on the adversaries.

Let K_a and $K_b \geqslant 1$ denote the number of malicious and benign devices, respectively, such that $K = K_a + K_b$.

Definition 11. *We say that a device k is a benign device if $w_k \sim \theta + \mathcal{N}(0, \tau^2\mathbf{I}_d)$; and we say a device k is a malicious device (or an adversary) if $w_k \sim \theta + \mathcal{N}(0, \tau_a^2\mathbf{I}_d)$ where $\tau_a > \tau$.*

As mentioned in Definition 1 and 2, in the presence of adversaries, we measure fairness as the performance variance on *benign* devices, and robustness as the average performance across *benign* devices. We next characterize the benefits of `Ditto` under such metrics.

Lemma 14. *Let w_k be the underlying model parameter of a benign device k. Let*

$$\widehat{w}_i := (X_i^T X_i)^{-1} X_i^T y_i,\ i \in [K]. \tag{5.55}$$

Let

$$\Sigma_w^{\setminus k} = \frac{1}{(K-1)^2}\left(\sum_{i\in[K_b],i\neq k}\left(\sigma^2(X_i^TX_i)^{-1}+\tau^2\mathbf{I}_d\right)+\sum_{i\in[K_a],i\neq k}\left(\sigma^2(X_i^TX_i)^{-1}+\tau_a^2\mathbf{I}_d\right)\right), \tag{5.56}$$

and

$$\Sigma_{w,a}^{-1} = (\sigma^2(X_k^TX_k)^{-1})^{-1}+(\Sigma_w^{\setminus k}+\tau^2\mathbf{I}_d)^{-1}. \tag{5.57}$$

Conditioned on observations $\widehat{w}_k$ and $\widehat{w}^{K\setminus k} := \frac{1}{K-1}\sum_{i\neq k,i\in[K]}\widehat{w}_i$, we have

$$w_k = \Sigma_{w,a}(\sigma^2(X_k^TX_k)^{-1})^{-1}\widehat{w}_k+\Sigma_{w,a}(\Sigma_w^{\setminus k}+\tau^2\mathbf{I}_d)^{-1}\widehat{w}^{K\setminus k}+\zeta_k, \tag{5.58}$$

where $\zeta_k \sim \mathcal{N}(0,\Sigma_{w,a})$.

Proof. For malicious devices $i\in[K_a]$ and $i\neq k$, the additive covariance of w_i with respect to θ is $\sigma^2(X_i^TX_i)^{-1}+\tau_a^2\mathbf{I}_d$. For benign devices $i\in[K_b]$ and $i\neq K$, the covariance is $\sigma^2(X_i^TX_i)^{-1}+\tau^2\mathbf{I}_d$. Therefore, the covariance of $\widehat{w}^{K\setminus k}$ is $\Sigma_w^{\setminus k}$. Hence given $\widehat{w}^{K\setminus k}$, w_k is Gaussian with covariance $\Sigma_w^{\setminus k}+\tau^2\mathbf{I}_d$. $\widehat{w}^{K\setminus k}$ can be viewed as a noisy observation of w_k with covariance $\Sigma_w^{\setminus k}+\tau^2\mathbf{I}_d$. $\widehat{w}_k$ is a noisy observation of w_k with covariance $\sigma^2(X_k^TX_k)^{-1}$. The proof follows by applying Lemma 9 to $\widehat{w}_k$ and $\widehat{w}^{K\setminus k}$. □

Theorem 8. *Assume for any $1\leqslant i\leqslant K$, $X_i^TX_I=\beta\mathbf{I}_d$ for some constant β. Let k be a benign device. Let λ_a^* be the optimal λ that minimizes the test performance on device k, i.e.,*

$$\lambda^* = \arg\min_\lambda E\left\{F_k(\widehat{w}_k(\lambda))|\,\widehat{w}_k,\widehat{w}^{K\setminus k}\right\}. \tag{5.59}$$

Then,

$$\lambda_a^* = \frac{\sigma^2}{n}\frac{K}{K\tau^2+\frac{K_a}{K-1}(\tau_a^2-\tau^2)}. \tag{5.60}$$

Proof. We obtain λ_a^* following the proof of Theorem 6. □

Theorem 9. *Among all `Ditto` solutions parameterized by λ, λ_a^* results in the most fair performance across all benign devices, i.e., it minimizes the variance of test performance (test mean square error) on benign devices.*

Proof. Similarly, we look at the variance of the test loss across benign devices:

$$\arg\min_\lambda E\left\{\mathbf{Var}_{K_b}\left\{\|X_k\widehat{w}_k(\lambda)-y_k\|_2^2\right\}\right\} \tag{5.61}$$

$$= \arg\min_\lambda E\left\{\mathbf{Var}_{K_b}\left\{\|w_k(\lambda)-w_k\|_2^2\right\}\right\} \tag{5.62}$$

$$= \arg\min_{\lambda} \widehat{E}_{K_b} \left\{ \left(\|w_k - \widehat{w}_k\|_2^2 \right)^2 \right\} - \left(\widehat{E}_{K_b} \left\{ \|w_k - \widehat{w}_k(\lambda)\|_2^2 \right\} \right)^2. \tag{5.63}$$

The rest of the proof is the same as the proof of Theorem 7, except that we set $a_k = \widehat{w}_k(\lambda) - \widehat{w}_k(\lambda_a^*)$. □

Remark 5. *For any benign device k, the solution we obtain by solving* `Ditto` *with λ_a^* is the most robust solution one could obtain among any federated point estimation method given observations $\widehat{w}_k$ and $\widehat{w}^{K\setminus k}$. λ_a^* also results in a most fair model in the solution space of* `Ditto` *parameterized by λ.*

Lemma 15. *The expected test error minimized at λ_a^* is $d\sigma_{w,a}^2$; and the variance of the test loss minimized at λ_a^* is $2d\sigma_{w,a}^4$, where $\sigma_{w,a}$ denotes the diagonal element of $\Sigma_{w,a}$.*

Proof. For the expected test performance, we note that

$$E\left\{ \|w_k - \widehat{w}_k(\lambda_a^*)\|^2 \middle| \widehat{w}^{K\setminus k}, \widehat{w}_k \right\} = E[\|\mathrm{diag}(\Sigma_{w,k})\|^2] = d\sigma_{w,k}^2. \tag{5.64}$$

For variance, as $a_k = 0$ if $\lambda = \lambda_a^*$, from (5.54), we get

$$\mathbf{Var}_{K_b} \left\{ \|w_k - \widehat{w}_k(\lambda_a^*)\|^2 \right\} = 2d\sigma_{w,k}^4. \tag{5.65}$$

□

Observations. From λ_a^*, we have the following interesting observations.

- Mean test error on benign devices (robustness) and variance of the performance across benign devices (fairness) can still be minimized with the same λ_a in the presence of adversaries.
- As $\tau_a \to \infty$, $\lambda_a^* \to 0$, i.e., training local models is optimal in terms of robustness and fairness when adversary's task may be arbitrarily far from the the task in the benign devices.
- As $\tau \to 0$, if $\tau_a > 0$, $\lambda_a^* < \infty$, which means that learning a global model is *not* optimal even with homogeneous data in the presence of adversaries.
- λ_a^* is a decreasing function of the number (K_a) and the capability (τ_a) of the corrupted devices. In other words, as the attacks become more adversarial, we need more personalization.
- The smallest test error is $\sigma_{w,a}^2$, and the optimal variance is $2\sigma_{w,a}^4$, which are both increasing with K_a (number of adversarial devices) or τ_a (the power of adversary) by inspecting (5.56) and (5.57). This reveals a fundamental tradeoff between fairness and robustness.

Discussion. Through our analysis, we prove that `Ditto` with an appropriate λ is more accurate, robust, and fair compared with training global or local models on the problem

described in 5.4.2. We provide closed-form solutions for λ^* across different settings (with and without adversaries), and show that `Ditto` can achieve fairness and robustness jointly. In the future, we plan to generalize the current theoretical framework to more general models. In the next section, we present a special case of the current analysis, a federated point estimation problem, which is also studied in Section 5.2.3 as a motivating example.

5.4.3 The Case of Federated Point Estimation

We consider the one-dimensional federated point estimation problem, which is a special case of linear regression. Similarly, Let θ be drawn from the non-informative prior on $\mathbb{R}$. We assume that K devices have their data distributed with parameters $\{w_k\}_{k\in[K]}$.

$$w_k = \theta + \zeta_k, \tag{5.66}$$

where $\zeta_k \sim \mathcal{N}(0, \tau^2)$ are IID.

Let each device have n data points denoted by $\mathbf{x}_k = \{x_{k,1}, \ldots, x_{k,n}\}$, such that

$$x_{k,i} = w_k + z_{k,i}, \tag{5.67}$$

where $z_{k,i} \sim \mathcal{N}(0, \sigma^2)$ and are IID.

Assume that

$$F_k(w) = \frac{1}{2}\left(w - \frac{1}{n}\sum_{i\in[n]} x_{k,i}\right)^2, \tag{5.68}$$

and denote by $\widehat{w}_k$ the minimizer of the empirical loss F_k. It is clear that

$$\widehat{w}_k = \frac{1}{n}\sum_{i\in[n]} x_{k,i}. \tag{5.69}$$

Further, let

$$w^* := \arg\min_w \left\{\frac{1}{K}\sum_{k\in[K]} F_k(w)\right\}. \tag{5.70}$$

It is straightforward calculation to verify that

$$w^* = \frac{1}{nK}\sum_{i\in[n]}\sum_{k\in[K]} x_{k,i} = \frac{1}{K}\sum_{k\in[K]} \widehat{w}_k. \tag{5.71}$$

Lemma 16. *Denote by $\widehat{w}_k(\lambda)$ the minimizer of h_k. Then,*

$$\widehat{w}_k(\lambda) = \frac{\lambda}{1+\lambda} w^* + \frac{1}{1+\lambda}\widehat{w}_k \tag{5.72}$$

$$= \frac{\lambda}{(1+\lambda)K}\sum_{j\neq k} \widehat{w}_j + \frac{K+\lambda}{(1+\lambda)K}\widehat{w}_k. \tag{5.73}$$

Let

$$\sigma_n^2 := \frac{\sigma^2}{n}, \tag{5.74}$$

and

$$\widehat{w}^{K\setminus k} := \frac{1}{K-1}\sum_{j\neq k}\widehat{w}_j. \tag{5.75}$$

Lemma 17. *Given observations $\widehat{w}^{K\setminus k}$ and $\widehat{w}_k$, w_k is Gaussian distributed and given by*

$$w_k = \frac{\sigma_w^2}{\sigma_n^2}\widehat{w}_k + \frac{(K-1)\sigma_w^2}{K\tau^2+\sigma_n^2}\widehat{w}^{K\setminus k} + \xi, \tag{5.76}$$

where

$$\frac{1}{\sigma_w^2} = \frac{1}{\sigma_n^2} + \frac{K-1}{K\tau^2+\sigma_n^2}, \tag{5.77}$$

and

$$\xi \sim \mathcal{N}\left(0, \sigma_w^2\right). \tag{5.78}$$

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Lemma 13. □

Theorem 10. *Let λ^* be the optimal λ that minimizes the test performance, i.e.,*

$$\lambda^* = \arg\min_{\lambda} E\left\{ (w_k - \widehat{w}_k(\lambda))^2 \middle| \widehat{w}^{K\setminus k}, \widehat{w}_k \right\}. \tag{5.79}$$

Then,

$$\lambda^* = \frac{\sigma_n^2}{\tau^2} = \frac{\sigma^2}{n\tau^2}. \tag{5.80}$$

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Theorem 6. □

Theorem 11. *Among all `Ditto`'s solutions, λ^* results in the most fair performance across all devices when there are no adversaries, i.e., it minimizes the variance of test performance (test mean square error).*

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Theorem 7. □

Similarly, the adversarial case presented below (including setups, lemmas, and theorems) is also a special case of the adversarial scenarios for linear regression.

Let K_a and $K_b \geqslant 1$ denote the number of adversarial and benign devices, respectively, such that $K = K_a + K_b$.

Definition 12. *We say that a device k is a benign device if $w_k \sim \theta + \mathcal{N}(0, \tau^2)$; and we say a device k is a malicious device (or an adversary) if $w_k \sim \theta + \mathcal{N}(0, \tau_a^2)$ where $\tau_a \geqslant \tau$.*

Lemma 18. *Let w_k be the parameter associated with a benign device. Given observations $\widehat{w}^{K\backslash k} := \frac{1}{K-1}\sum_{j\neq k}\widehat{w}_j$ and $\widehat{w}_k$, w_k is Gaussian distributed and given by*

$$w_k = \frac{\sigma_{w,a}^2}{\sigma_n^2}\widehat{w}_k + \frac{(K-1)\sigma_{w,a}^2}{K\tau^2+\sigma_n^2+\frac{K_a}{K-1}(\tau_a^2-\tau^2)}\widehat{w}^{K\backslash k} + \xi_a, \tag{5.81}$$

where

$$\frac{1}{\sigma_{w,a}^2} = \frac{1}{\sigma_n^2} + \frac{K-1}{K\tau^2+\sigma_n^2+\frac{K_a}{K-1}(\tau_a^2-\tau^2)}, \tag{5.82}$$

and

$$\xi_a \sim \mathcal{N}\left(0, \sigma_{w,a}^2\right). \tag{5.83}$$

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Lemma 14. □

Theorem 12. *Let w_k be a benign device. Let λ_a^* be the optimal λ that minimizes the test performance, i.e.,*

$$\lambda_a^* = \arg\min_{\lambda} E\left\{ (w_k - \widehat{w}_k(\lambda))^2 \middle| \widehat{w}^{K\backslash k}, \widehat{w}_k \right\}. \tag{5.84}$$

Then,

$$\lambda_a^* = \frac{\sigma^2}{n}\frac{K}{K\tau^2+\frac{K_a}{K-1}(\tau_a^2-\tau^2)}. \tag{5.85}$$

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Theorem 8. □

Theorem 13. *Among all solutions of Objective* (`Ditto`) *parameterized by λ, λ_a^* results in the most fair performance across all benign devices, i.e., it minimizes the variance of test performance (test mean square error) on benign devices.*

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Theorem 9. □

Lemma 19. *The expected test error minimized at λ_a^* is $\sigma_{w,a}^2$; and the variance of the test performance minimized at λ_a^* is $2\sigma_{w,a}^4$.*

Proof. The proof follows by setting $X_k = \mathbf{1}_{n\times 1}$ ($k \in [K]$) in Lemma 15. □

5.5 Algorithm and Convergence Analysis

In this section, we first present the specific algorithm (Algorithm 7) that we use for most of our experiments (all except for Table 5.3 and 5.6). Algorithm 7 is a special case of the more general `Ditto` solver (Algorithm 6), where we use $\min_w \sum_{k\in[K]} p_k F_k(w)$ as the global objective and FedAvg as its solver. As before, the `Ditto` personalization add-on is highlighted in red. In addition, we prove that personalized models can inherit the convergence rates of the optimal global model for any $G(\cdot)$ (Theorem 14), and provide convergence guarantees for the special case of Algorithm 7 (Corollary 4).

Algorithm 7 `Ditto` for Personalized FL in the case of $G(\cdot)$ being FedAvg [206]

Input: K, T, s, λ, η_g, η_l, w^0, p_k, $\{v_k^0\}_{k\in[K]}$
for $t = 0, \cdots, T-1$ **do**
  Server randomly selects a subset of devices S_t, and sends w^t to them
  for device $k \in S_t$ in parallel **do**
    Sets w_k^t to w^t and updates w_k^t for r local iterations on F_k:

$$w_k^t = w_k^t - \eta_g \nabla F_k(w_k^t)$$

    Updates v_k for s local iterations:

$$v_k = v_k - \eta_l(\nabla F_k(v_k) + \lambda(v_k - w^t))$$

    Sends $\Delta_k^t := w_k^t - w^t$ back
  end for
  Server updating w^{t+1} as

$$w^{t+1} \leftarrow w^t + \frac{1}{|S_t|}\sum_{k\in S_t} \Delta_k^t$$

end for
return $\{v_k\}_{k\in[K]}$ (personalized), w^T (global)

To analyze the convergence behavior of Algorithm 6 and 7, we first state a list of assumptions below.

- The global model converges with rate $g(t)$, i.e., there exists $g(t)$ such that $\lim_{t\to\infty} g(t) = 0$, $\mathbb{E}[\|w^t - w^*\|^2] \leqslant g(t)$.
- For $k \in [K]$, F_k is μ-strongly convex.
- The expectation of stochastic gradients is uniformly bounded at all devices and all iterations, i.e.,

$$\mathbb{E}[\|\nabla F_k(w^t, \xi^t)\|^2] \leqslant G_1^2. \tag{5.86}$$

Let w^* be defined as

$$w^* := \min_w \; G(F_1(w), \dots \; F_K(w)) \tag{5.87}$$

i.e., w^* is the empirically optimal global model for $G(\cdot)$. Let u_k^* denote the empirically optimal local model on device k, i.e.,

$$u_k^* = \arg\min_u F_k(u). \tag{5.88}$$

We introduce an additional assumption on the distance between optimal local models $\{u_k^*\}_{k\in[K]}$ and the optimal global model w^* below.

- The L_2 distance between the optimal local models and the optimal global model is bounded, i.e., for $k \in [K]$,

$$\|u_k^* - w^*\| \leqslant M. \tag{5.89}$$

This assumption sets an upper bound on the deviation of the local model on device k, with the global model. It can in turn be viewed as boundedness of heterogeneity of the training data across devices. When local data are farther from being IID, M tends to be larger. Recall that in the fairness/robustness analysis of `Ditto` (Appendix 5.4), we model the relatedness of *underlying* models via τ, and $\mathbb{E}[\|w_k - \theta\|^2] = d\tau^2$ where w_k is the *underlying* model for device k and d is the model dimension. M is related to τ^2 as

$$\begin{aligned}\mathbb{E}[\|u_k^* - w^*\|^2] &\leqslant 2\mathbb{E}[\|\mu_k^* - w_k\|^2] + 4\mathbb{E}[\|w_k - \theta\|^2] + 4\mathbb{E}[\|\theta - w^*\|^2] &(5.90)\\ &\to 4d\tau^2. &(5.91)\end{aligned}$$

when n_k and the total number of samples across all devices are sufficiently large, considering the linear problems we studied. We later show that for convergence, λ scales with $1/M$, which is consistent with λ^* (for fairness/robustness) scaled with $1/\tau^2$.

Further let

$$v_k^* = \arg\min_v h_k(v; w^*), \tag{5.92}$$

i.e., v_k^* is the optimal personalized model for device k. We are interested in the convergence of v_k to v_k^*. We first characterize the progress of updating personalized models for one step under a *general* $G(\cdot)$.

Lemma 20 (Progress of one step)**.** *Under assumptions above, let device k get selected with probability p_k at each communication round, with decaying local step-size $\frac{2}{(t+1)(\mu+\lambda)p_k}$, at each communication round t, we have*

$$\begin{aligned}\mathbb{E}[\|v_k^{t+1} - v_k^*\|^2] \leqslant \left(1 - \frac{2}{t+1}\right)\mathbb{E}[\|v^t - v^*\|^2] + \frac{4(G_1 + \lambda(M + \frac{G_1}{\mu}))^2}{(t+1)^2(\mu+\lambda)^2 p_k^2}\\ + \frac{4\lambda^2}{(t+1)^2(\mu+\lambda)^2 p_k^2}\mathbb{E}[\|w^t - w^*\|^2]\end{aligned}$$

$$+\frac{8\lambda(G_1+\lambda(M+\frac{G_1}{\mu}))}{(t+1)^2(\mu+\lambda)^2p_k^2}\sqrt{\mathbb{E}[\|w^t-w^*\|^2]}$$
$$+\frac{4\lambda}{(t+1)(\mu+\lambda)p_k}\sqrt{\mathbb{E}[\|v_k^t-v_k^*\|^2]\mathbb{E}[\|w^t-w^*\|^2]}. \tag{5.93}$$

Proof. Denote $g(v_k^t;w^t)$ as the stochastic gradient of $h_k(v_k^t;w^t)$. Let I_t indicate if device k is selected at the t-th round, and $\mathbb{E}[I_t]=p_k$.

$$\mathbb{E}[\|v_k^{t+1}-v_k^*\|^2]=\mathbb{E}[\|v_k^t-\eta I_t g(v_k^t;w^t)-v_k^*\|^2] \tag{5.94}$$
$$=\mathbb{E}[\|v_k^t-v_k^*\|^2]+\eta^2\mathbb{E}[\|I_t g(v_k^t;w^t)\|^2]+2\eta\mathbb{E}\langle I_t g(v_k^t;w^t),v_k^*-v_k^t\rangle \tag{5.95}$$
$$\leqslant(1-(\mu+\lambda)\eta p_k)\mathbb{E}[\|v_k^t-v_k^*\|^2]+\eta^2\mathbb{E}[\|g(v_k^t;w^t)\|^2]$$
$$+2\eta p_k\mathbb{E}[h(v_k^*;w^t)-h(v_k^t;w^t)] \tag{5.96}$$
$$\leqslant(1-(\mu+\lambda)\eta p_k)\mathbb{E}[\|v_k^t-v_k^*\|^2]$$
$$+\eta^2\mathbb{E}[\|g(v_k^t;w^*)\|^2]+\eta^2\lambda^2\mathbb{E}[\|w^t-w^*\|^2]+2\eta^2\lambda\mathbb{E}[\|g(v_k^t;w^*)\|\|w^t-w^*\|]$$
$$+2\eta p_k(h(v_k^*;w^*)-\mathbb{E}[h(v_k^t;w^*)])+2\eta p_k\lambda\mathbb{E}[\|v_k^t-v_k^*\|\|w^t-w^*\|]. \tag{5.97}$$

Further, note that

$$\mathbb{E}[\|v_k^t-u_k^*\|^2]\leqslant\frac{1}{\mu^2}\mathbb{E}[\|\nabla F_k(v_k^t)\|^2]\leqslant\frac{G_1^2}{\mu^2}, \tag{5.98}$$
$$\mathbb{E}[\|v_k^t-w^*\|^2]=\mathbb{E}[\|v_k^t-u_k^*+u_k^*-w^*\|^2] \tag{5.99}$$
$$\leqslant\mathbb{E}[\|v_k^t-u_k^*\|^2]+\mathbb{E}[\|u_k^*-w^*\|^2]+2\mathbb{E}[\|v_k^t-u_k^*\|\|u_k^*-w^*\|] \tag{5.100}$$
$$\leqslant\frac{G_1^2}{\mu^2}+M^2+\frac{2MG_1}{\mu}, \tag{5.101}$$
$$\mathbb{E}[\|g(v_k^t;w^*)\|^2]=\mathbb{E}[\|\nabla F_k(v_k^t)+\lambda(v_k^t-w^*)\|^2] \tag{5.102}$$
$$\leqslant G_1^2+\lambda^2(\frac{G_1}{\mu}+M)^2+2G_1\lambda(\frac{G_1}{\mu}+M). \tag{5.103}$$

Plug it into Eq. (5.97),

$$\mathbb{E}[\|v_k^{t+1}-v_k^*\|^2]$$
$$\leqslant(1-(\mu+\lambda)\eta p_k)\mathbb{E}[\|v_k^t-v_k^*\|^2]+\eta^2(G_1+\lambda(M+\frac{G_1}{\mu}))^2+\eta^2\lambda^2\mathbb{E}[\|w^t-w^*\|^2]$$
$$+2\eta^2\lambda(G_1+\lambda(M+\frac{G_1}{\mu}))\sqrt{\mathbb{E}[\|w^t-w^*\|^2]}+2\eta p_k\lambda\sqrt{\mathbb{E}[\|v_k^t-v_k^*\|^2]\mathbb{E}[\|w^t-w^*\|^2]}.$$

where the last step is due to $E[XY]\leqslant\sqrt{E[X^2]E[Y^2]}$. The Lemma then holds by taking $\eta=\frac{2}{(t+1)(\mu+\lambda)p_k}$. □

Lemma 20 relates $\mathbb{E}[\|v_k^{t+1} - v_k^*\|^2]$ with $\mathbb{E}[\|v_k^t - v_k^*\|^2]$ and $\mathbb{E}[\|w_k^t - w^*\|^2]$. Based on this, we prove that personalized models can inherit the convergence rate of the global model w^t for any $G(\cdot)$.

Theorem 14 (Relations between convergence of global and personalized models)**.** *Under the assumptions above, if there exists a constant A such that $\frac{g(t+1)}{g(t)} \geqslant 1 - \frac{g(t)}{A}$, then there exists $C < \infty$ such that for any device $k \in [K]$, $\mathbb{E}[\|v_k^t - v_k^*\|^2] \leqslant Cg(t)$ with a local learning rate $\eta = \frac{2g(t)}{A(\mu+\lambda)p_k}$.*

Proof. We proceed the proof by induction. First, for any constant $C > \frac{\mathbb{E}[\|v_k^0 - v_k^*\|^2]}{g(0)}$, $\mathbb{E}[\|v_k^0 - v_k^*\|^2] \leqslant Cg(0)$. If $\mathbb{E}[\|v_k^t - v_k^*\|^2] \leqslant Cg(t)$ holds, then for $t+1$, from Lemma 20,

$$\begin{aligned}
\mathbb{E}[\|v_{k+1}^t - v_k^*\|^2] \quad &\leqslant \left(1 - \frac{2g(t)}{A}\right) Cg(t) \\
&\quad + \frac{g(t)^2}{A} \frac{4}{Ap_k^2} \left(\frac{(G_1 + \lambda(M + \frac{G_1}{\mu}))^2}{(\mu+\lambda)^2} + g(t) + \frac{2(G_1 + \lambda(M + \frac{G_1}{\mu}))\sqrt{g(t)}}{\mu+\lambda} \right) && (5.104) \\
&\quad + g(t)^2 \frac{4\lambda\sqrt{C}}{(\mu+\lambda)} && (5.105) \\
&\leqslant \left(1 - \frac{2g(t)}{A}\right) Cg(t) + \frac{Cg(t)^2}{A} && (5.106)
\end{aligned}$$

holds for some $C < \infty$. Hence,

$$\begin{aligned}
\mathbb{E}[\|v_{k+1}^t - v_k^*\|^2] &\leqslant \left(1 - \frac{2g(t)}{A}\right) Cg(t) + \frac{Cg(t)^2}{A} && (5.107) \\
&= \left(1 - \frac{g(t)}{A}\right) Cg(t) && (5.108) \\
&\leqslant Cg(t+1), && (5.109)
\end{aligned}$$

completing the proof. □

Discussions. Theorem 14 also suggests how the percentage/power of malicious devices can affect convergence rates. The percentage/power of adversaries impacts both the optimal global solution w^*, and the convergence rate of the global model $g(t)$. (i) For w^*, it affects M in Eq (5.89)—the distance between the local model on a benign device and the global model. This in turn affects λ in Eq (5.93) and (5.104), and the constant C. λ can scale inversely proportional to M, which is consistent with our fairness/robustness analysis where λ^* should decrease as the increase of τ^2. (ii) For $g(t)$, the modularity

of Ditto allows for decoupling the convergence of personalized models and the global model (as demonstrated by this theorem), and we can plug in any previous algorithms and their analysis on the convergence rate $g(t)$ as a function of malicious devices.

As a direct result of Theorem 14, we could state a result for `Ditto` when the global objective is FedAvg.

Corollary 4 (Convergence of personalized models)**.** *Under the assumptions above, if the global objective $G(\cdot)$ is FedAvg, then under Algorithm 7, for $k \in [K]$,*

$$\mathbb{E}[\|v_k^t - v_k^*\|^2] = O(1/t). \tag{5.110}$$

Proof. From Li et al. [189] Theorem 2, we know the global model for FedAvg converges at a rate of $O(1/t)$, i.e.,

$$\mathbb{E}[\|w^t - w^*\|^2] \leqslant \frac{D'}{t+B}\|w^1 - w^*\|^2 \leqslant \frac{D}{t+1}, \tag{5.111}$$

where D, D', B are constants. Setting $g(t) = \frac{D}{t+1}$ and $A = D$ in Theorem 14, it follows that $\mathbb{E}[\|v_k^t - v_k^*\|^2] = O(1/t)$. □

5.6 Experimental Details

5.6.1 Datasets and Models

We summarize the datasets, corresponding models, and tasks in Table 5.4 below. We evaluate the performance of `Ditto` with both convex and non-convex models across a set of FL benchmarks. In our datasets, we have both image data (FEMNIST, CelebA, Fashion MNIST), and text data (StackOverflow).

Table 5.4: Summary of datasets.

Datasets	# Devices	Data Partitions	Models	Tasks
Vehicle [75][2]	23	natural (each device is a vehicle)	linear SVM	binary classification
FEMNIST [54]	205	natural (each device is a writer)	CNN	62-class classification
CelebA [195]	515	natural (each device is a celebrity)	CNN	binary classification
Fashion MNIST [302]	500	synthetic (assign 5 classes to each device)	CNN	10-class classification
StackOverflow [1][3]	400	natural (each device is a user)	logistic regress.	500-class tag prediction
FEMNIST (skewed) [54]	100	synthetic (assign 5 classes to each device)	CNN	62-class classification

FEMNIST is Federated EMNIST, which is EMNIST [54] partitioned by the writers of digits/characters created by a previous federated learning benchmark [43]. We have two versions of FEMNIST in this work under different partitions with different levels of statistical heterogeneity. The manually-partitioned version is more heterogeneous than the naturally-partitioned one, as we assign 5 classes to each device. We show that the benefits of `Ditto` can be more significant on the skewed FEMNIST data (Table 5.10). All results shown in the main text are based on the natural partition. We downsample the number of data points on each device (following the power law) for Vehicle. For FEMNIST, CelebA, and StackOverflow, we randomly sample devices (users) from the entire dataset. We use the full version of Fashion MNIST (which has been used in previous FL works [29]), and assign 5 classes to each device.

5.6.2 Personalization Baselines

We elaborate on the personalization baselines used in our experiments (Table 5.2) which allow for partial device participation and local updating. We consider:

- **MOCHA** [265], a primal-dual framework for multi-task learning. It jointly learns the model parameters and a device relation matrix, and applicable to convex problems.
- **APFL** [69], which proposes to interpolate between local and global models for personalization. While it can reduce to solving local problems (without constraints on the solution space) as pointed out in [69], we find that in neural network applications, it has some personalization benefits, possibly due to the joint optimization solver.

[2] `http://www.ecs.umass.edu/~mduarte/Software.html`

[3] `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data`.

- **Elastic Weight Consolidation (EWC)**, which takes into account the Fisher information when finetuning from the optimal global model [156, 312]. The local objective is $\min_w F_k(w) + \frac{\lambda}{2} \sum_i \mathbf{F}_{ii} \cdot (w[i] - w^*[i])^2$ where $[i]$ denotes the index of parameters and $\mathbf{F}_{ii}$ denotes the i-th diagonal of the empirical Fisher matrix $\mathbf{F}$ estimated using a data batch.
- **L2SGD**, which regularizes personalized models towards their mean [107]. The proposed method requires full device participation once in a while. However, to remain consistent with the other solvers, we use their objective but adopt a different solver with partial device participation—each selected local device solving $\min_w F_k(w) + \frac{\lambda}{2}\|w - \bar{w}\|^2$ where $\bar{w}$ is the current mean of all personalized models $\bar{w} = \frac{1}{N} \sum_{k=1}^{N} w_k$.
- **Mapper**, which is one of the three personalization methods proposed in Mansour et al. [202] that needs the minimal amount of meta-information. Similar to APFL, it is also motivated by model interpolation.
- **Per-FedAvg** (HF) [88] which applies MAML [92] to personalize federated models with an Hessian-product approximation to approximate the second-order gradients.
- **Symmetrized KL** constrains the symmetrized KL divergence between the prediction of finetuned models and that of the initialization. Specifically, in our setting, the local objective is $\min_w F_k(w) + \frac{\lambda}{2} \left(D_{\mathrm{KL}}(f(w)||f(w^*)) + D_{\mathrm{KL}}(f(w^*)||f(w))\right)$ where $D_{\mathrm{KL}}(P||Q)$ is the KL-divergence between P and Q, and $f(\cdot)$ denotes the softmax probability for classification.

5.7 Additional and Complete Experiment Results

5.7.1 Comparing with Finetuning

As discussed in Section 7.5, finetuning on h_k for each device k is a possible solver for `Ditto`. In non-convex cases, however, starting from a corrupted w^* may result in inferior performance compared with Algorithm 6. We provide a simple example to illustrate this point. To perform finetuning, we run different numbers of epochs of mini-batch SGD on the `Ditto` objective for each device in the network, and pick the best one. As shown in Figure 5.7 below, finetuning at round 5,000 will not result in a good final accuracy. We observe that one could also stop at early iterations and then finetune. However, it is difficult to do so in practice based on the training or validation data alone, as shown in Figure 5.8.

5.7.2 Tuning λ

We assume that the server *does not* have knowledge of which devices are benign vs. malicious, and we have each device *locally* select and apply a best λ from a candidate set of three values based on their validation data. For benign devices, this means they

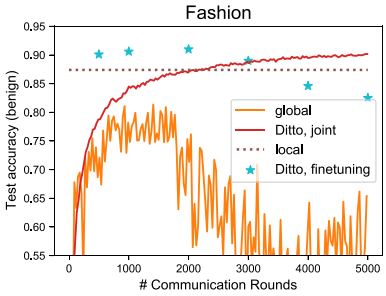


Figure 5.7: 'Ditto, joint' achieves high test accuracy on benign devices. The performance can also be good if we first early stop at some specific points and then finetune.

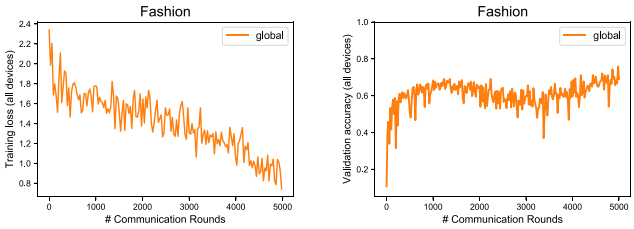


Figure 5.8: Finetuning is not very practical as it is difficult to determine when to stop training the global model by looking at the training loss (left) or validation accuracy (right) on all devices (without knowing which are benign).

will pick a λ based on their clean validation signal. For malicious devices, how they perform personalization (i.e., selecting λ) does not affect the corrupted global model updates they send, which are independent of λ. We further assume the devices have some knowledge of how 'strong' the attack is. We define strong attacks as (i) all of model replacement attacks (A3) where the magnitude of the model updates from malicious devices can scale by $> 10\times$, and (ii) other attacks where more than half of the devices are corrupted. In particular, for devices with very few validation samples (less than 4), we use a fixed small λ (λ=0.1) for strong attacks, and use a fixed relatively large λ (λ=1) for all other attacks. For devices with more than 5 validation data points, we let each select λ from $\{0.05, 0.1, 0.2\}$ for strong attacks, and select λ from $\{0.1, 1, 2\}$ for all other attacks. For the StackOverflow dataset, we tune λ from $\{0.01, 0.05, 0.1\}$ for strong attacks, and $\{0.05, 0.1, 0.3\}$ for all other attacks. We directly evaluate our hyperparameter tuning strategy in Table 5.5 below—showing that this dynamic tuning heuristic works well relative to an ideal, but more unrealistic strategy that picks the best λ based on knowledge of which devices are benign vs. malicious (i.e., by only using the validation data of the benign devices).

Table 5.5: Results (test accuracy and standard deviation) of using dynamic λ's. 'Best λ' refers to the results of selecting the best (fixed) λ based on average validation performance on benign devices (assuming the server knows which devices are malicious).

FEMNIST		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
best λ	0.836 (.10)	0.803 (.10)	0.767 (.10)	0.672 (.14)	0.792 (.11)	0.743 (.14)	0.674 (.14)	0.691 (.15)	0.664 (.14)	0.650 (.14)
dynamic λ's	0.834 (.09)	0.802 (.10)	0.762 (.11)	0.672 (.13)	0.801 (.09)	0.700 (.15)	0.675 (.14)	0.685 (.15)	0.650 (.14)	0.613 (.13)
Fashion		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
best λ	0.946 (.06)	0.944 (.08)	0.935 (.07)	0.925 (.07)	0.943 (.08)	0.930 (.07)	0.912 (.08)	0.914 (.09)	0.903 (.09)	0.873 (.09)
dynamic λ's	0.943 (.06)	0.944 (.07)	0.937 (.07)	0.907 (.10)	0.938 (.07)	0.930 (.08)	0.913 (.09)	0.921 (.09)	0.902 (.09)	0.872 (.11)
CelebA		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
best λ	0.914 (.18)	0.828 (.22)	0.721 (.27)	0.724 (.28)	0.872 (.22)	0.826 (.26)	0 708 (.29)	0.699 (.28)	0.694 (.27)	0.689 (.28)
dynamic λ's	0.911 (.16)	0.820 (.26)	0.714 (.28)	0.724 (.28)	0.872 (.22)	0.826 (.26)	0.706 (.28)	0.699 (.28)	0.694 (.27)	0.689 (.28)
Vehicle		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
best λ	0.882 (.05)	0.862 (.05)	0.841 (.09)	0.851 (.06)	0.884 (.05)	0.872 (.06)	0.879 (.04)	0.872 (.06)	0.829 (.08)	0.827 (.08)
dynamic λ's	0.872 (.05)	0.857 (.06)	0.827 (.08)	0.834 (.05)	0.872 (.06)	0.867 (.07)	0.848 (.04)	0.839 (.08)	0.824 (.08)	0.822 (.09)
StackOverflow		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
best λ	0.315 (.16)	0.325 (.16)	0.315 (.17)	0.313 (.15)	0.314 (.16)	0.350 (.16)	0.312 (.14)	0.316 (.17)	0.321 (.17)	0.327 (.17)
dynamic λ's	0.317 (.17)	0.323 (.18)	0.314 (.16)	0.359 (.16)	0.326 (.17)	0.317 (.17)	0.301 (.17)	0.318 (.17)	0.319 (.17)	0.311 (.17)

5.7.3 `Ditto` Augmented with Robust Baselines

In Section 5.3.4, we demonstrate that the performance of `Ditto` can be further improved when it is combined with robust baselines (e.g., learning a robust w^* via robust aggregation). Here, we report full results validating this claim in Table 5.6 below.

Table 5.6: `Ditto` augmented with robust baselines (full results).

FEMNIST	**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.773 (.11)	0.727 (.12)	0.574 (.15)	0.774 (.11)	0.703 (.14)	0.636 (.15)	0.517 (.14)	0.487 (.14)	0.364 (.13)
clipping	0.791 (.11)	0.736 (.11)	0.408 (.14)	0.791 (.11)	0.736 (.13)	0.656 (.13)	0.795 (.11)	0.060 (.05)	0.061 (.05)
`Ditto`	0.803 (.10)	**0.767 (.10)**	**0.672** (.14)	0.792 (.11)	0.743 (.14)	0.674 (.14)	0.691 (.15)	0.664 (.14)	0.650 (.14)
`Ditto` + clipping	**0.810 (.11)**	0.762 (.11)	0.645 (.13)	**0.808 (.11)**	**0.757 (.11)**	**0.684 (.13)**	**0.813 (.13)**	**0.707 (.15)**	**0.672 (.14)**
CelebA	**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.810 (.22)	0.535 (.26)	0.228 (.21)	0.869 (.22)	0.823 (.23)	0.656 (.26)	0.451 (.27)	0.460 (.29)	0.515 (.31)
multi-Krum	**0.882 (.22)**	0.564 (.26)	0.107 (.19)	0.887 (.21)	0.891 (.20)	0.617 (.30)	0.512 (.27)	0.529 (.27)	0.430 (.26)
`Ditto`	0.828 (.22)	0.721 (.27)	0.724 (.28)	0.872 (.22)	0.826 (.26)	0.708 (.29)	0.699 (.28)	0.694 (.27)	0.689 (.28)
`Ditto` + multi-Krum	0.875 (.20)	**0.722 (.26)**	**0.733 (.27)**	**0.903 (.20)**	**0.902 (.21)**	**0.885 (.23)**	**0.713 (.28)**	**0.709 (.28)**	**0.713 (.28)**

5.7.4 `Ditto` Complete Results

In Section 5.3.1, we present partial results on three strong attacks on two datasets. Here, we provide full results showing the robustness and fairness of `Ditto` on all attacks and all datasets compared with all defense baselines. We randomly split local data on each device into 72% train, 8% validation, and 20% test sets, and report all results on test data. We use a learning rate of 0.01 for StackOverflow, 0.05 for Fashion MNIST and 0.1 for all other datasets; and batch size 16 for CelebA and Fashion MNIST, 32 for FEMNIST and Vehicle, and 100 for StackOverflow. For every dataset, we first run FedAvg on clean data to determine the number of communication rounds. Then we run the same number of rounds for all attacks on that dataset.

For our robust baselines, 'median' means coordinate-wise median. For Krum, multi-Krum, k-norm, and k-loss, we assume the server knows the expected number of malicious devices when aggregation. In other words, for k-norm, we filter out the updates with the k largest norms where k is set to the expected number of malicious devices. Similarly, for k-loss, we only use the model update with the k+1-th largest training loss. For gradient clipping, we set the threshold to be the median of the gradient norms coming from all selected devices at each round. FedMGDA+ has an additional ϵ hyperparameter which we select from $\{0, 0.1, 0.5, 1\}$ based on the validation performance on benign devices. For the finetuning (only on neural network models) baseline, we run 50 epochs of mini-batch SGD on each device on the local objective F_k starting from w^*. We see that `Ditto` can achieve better fairness and robustness in most cases. In particular, on average of all datasets and all attack scenarios, `Ditto` (with dynamic λ's) achieves 6% absolute accuracy improvement compared with the strongest robust baseline. In terms of fairness, `Ditto` is able to reduce the variance of test accuracy by 10% while improving the average accuracy by 5% relative to state-of-the-art methods for fair FL (without attacks).

Table 5.7: Full results (average and standard deviation of test accuracy across all devices) on the Vehicle dataset with linear SVM. On this convex problem, we additionally compare with another primal-dual MTL method MOCHA [265], which suggests the fairness/robustness benefits of other MTL approaches.

Vehicle		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
global	0.866 (.16)	0.847 (.08)	0.643 (.10)	0.260 (.27)	0.866 (.18)	0.840 (.21)	0.762 (.27)	0.854 (.17)	0.606 (.08)	0.350 (.19)
local	0.836 (.07)	0.835 (.08)	0.840 (.09)	**0.857 (.09)**	0.835 (.08)	0.840 (.09)	0.857 (.09)	0.840 (.07)	0.835 (.08)	**0.840 (.09)**
fair	0.870 (.08)	0.721 (.06)	0.572 (.08)	0.404 (.13)	0.746 (.12)	0.704 (.15)	0.706 (.20)	0.775 (.13)	0.628 (.25)	0.448 (.11)
median	0.863 (.16)	0.861 (.18)	0.676 (.11)	0.229 (.31)	0.864 (.18)	0.838 (.21)	0.774 (.28)	0.867 (.17)	0.797 (.07)	0.319 (.17)
Krum	0.852 (.17)	0.853 (.19)	0.830 (.22)	0.221 (.32)	0.851 (.19)	0.828 (.22)	0.780 (.31)	0.867 (.17)	**0.866 (.18)**	0.588 (.14)
multi-Krum	0.866 (.16)	0.867 (.18)	0.839 (.20)	0.220 (.32)	0.867 (.18)	0.839 (.22)	0.770 (.31)	0.868 (.17)	0.836 (.08)	0.406 (.15)
clipping	0.864 (.16)	0.865 (.17)	0.678 (.34)	0.234 (.30)	0.865 (.18)	0.839 (.22)	0.764 (.27)	0.868 (.17)	0.789 (.07)	0.315 (.17)
k-norm	0.866 (.16)	**0.867 (.17)**	0.838 (.21)	0.222 (.32)	0.867 (.18)	0.839 (.22)	0.778 (.31)	0.867 (.17)	0.844 (.09)	0.458 (.16)
k-loss	0.850 (.05)	0.755 (.03)	0.732 (.09)	0.217 (.31)	0.852 (.06)	0.840 (.07)	0.825 (.09)	0.866 (.17)	0.692 (.08)	0.328 (.16)
FedMGDA+	0.860 (.16)	0.835 (.09)	0.674 (.14)	0.270 (.26)	0.860 (.18)	0.843 (.22)	0.794 (.26)	0.836 (.17)	0.757 (.07)	0.676 (.17)
MOCHA	0.880 (.04)	0.848 (.07)	0.832 (.08)	0.829 (.10)	0.846 (.06)	0.843 (.07)	0.833 (.10)	0.862 (.06)	0.844 (.07)	0.834 (.07)
`Ditto`, λ=0.1	0.845 (.07)	0.841 (.08)	**0.841 (.09)**	0.851 (.06)	0.844 (.07)	0.848 (.08)	0.866 (.05)	0.838 (.07)	0.829 (.08)	0.827 (.08)
`Ditto`, λ=1	0.875 (.05)	0.859 (.06)	0.821 (.07)	0.776 (.08)	0.875 (.06)	0.870 (.07)	**0.879 (.04)**	0.860 (.07)	0.813 (.07)	0.757 (.08)
`Ditto`, λ=2	**0.882 (.05)**	0.862 (.05)	0.800 (.07)	0.709 (.12)	**0.884 (.05)**	**0.872 (.06)**	0.869 (.04)	**0.872 (.06)**	0.791 (.06)	0.690 (.09)

Table 5.8: Full results (average and standard deviation of test accuracy across all devices) on FEMNIST.

FEMNIST		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.804 (.11)	0.773 (.11)	0.727 (.12)	0.574 (.15)	0.774 (.11)	0.703 (.14)	0.636 (.15)	0.517 (.14)	0.487 (.14)	0.364 (.13)
local	0.628 (.15)	0.620 (.14)	0.627 (.14)	0.607 (.13)	0.620 (.14)	0.627 (.14)	0.607 (.13)	0.622 (.14)	0.621 (.14)	0.620 (.14)
fair	0.809 (.11)	0.636 (.15)	0.562 (.13)	0.478 (.12)	0.440 (.15)	0.336 (.12)	0.363 (.12)	0.353 (.12)	0.316 (.12)	0.299 (.11)
median	0.733 (.14)	0.627 (.15)	0.576 (.15)	0.060 (.04)	0.673 (.14)	0.645 (.14)	0.564 (.15)	0.628 (.14)	0.573 (.15)	0.577 (.16)
Krum	0.717 (.16)	0.059 (.05)	0.096 (.07)	0.091 (.07)	0.604 (.14)	0.062 (.25)	0.024 (.02)	0.699 (.15)	0.719 (.13)	0.648 (.14)
multi-Krum	0.804 (.11)	0.790 (.11)	0.759 (.11)	0.115 (.07)	0.789 (.11)	0.762 (.11)	0.014 (.02)	0.529 (.14)	0.664 (.15)	0.561 (.14)
clipping	0.805 (.11)	0.791 (.11)	0.736 (.11)	0.408 (.14)	0.791 (.11)	0.736 (.13)	0.656 (.13)	0.795 (.11)	0.060 (.05)	0.061 (.05)
k-norm	0.806 (.11)	0.785 (.11)	0.760 (.12)	0.060 (.05)	0.788 (.10)	**0.765 (.11)**	0.011 (.02)	0.060 (.04)	0.647 (.15)	0.562 (.15)
k-loss	0.762 (.11)	0.606 (.13)	0.599 (.13)	0.596 (.13)	0.432 (.12)	0.508 (.13)	0.572 (.14)	0.060 (.04)	0.009 (.02)	0.006 (.01)
FedMGDA+	0.803 (.12)	0.794 (.12)	0.730 (.12)	0.057 (.04)	**0.793 (.12)**	0.753 (.12)	0.671 (.14)	**0.798 (.11)**	**0.794 (.12)**	**0.791 (.11)**
finetuning	0.815 (.09)	0.778 (.11)	0.734 (.12)	**0.671 (.13)**	0.764 (.11)	0.695 (.18)	0.646 (.14)	0.688 (.13)	0.671 (.14)	0.655 (.13)
`Ditto`, λ=0.01	0.800 (.15)	0.709 (.15)	0.683 (.17)	0.642 (.13)	0.701 (.14)	0.684 (.14)	0.645 (.14)	0.650 (.14)	0.628 (.14)	0.650 (.14)
`Ditto`, λ=0.1	0.827 (.10)	0.794 (.11)	0.755 (.13)	0.666 (.14)	0.786 (.13)	0.743 (.14)	**0.674 (.14)**	0.691 (.15)	0.664 (.14)	0.640 (.14)
`Ditto`, λ=1	**0.836 (.10)**	**0.803 (.10)**	**0.767 (.10)**	**0.672 (.14)**	**0.792 (.11)**	0.691 (.17)	0.575 (.17)	0.642 (.12)	0.595 (.14)	0.554 (.15)

Table 5.9: Full results (average and standard deviation of test accuracy across all devices) on Fashion MNIST.

Fashion MNIST		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	20%	50%
global	0.911 (.08)	0.897 (.08)	0.855 (.10)	0.753 (.13)	0.900 (.08)	0.882 (.09)	0.857 (.10)	0.753 (.10)	0.551 (.13)	0.275 (.12)
local	0.876 (.10)	0.874 (.10)	0.876 (.11)	0.879 (.10)	0.874 (.10)	0.876 (.11)	0.879 (.10)	0.877 (.10)	0.874 (.10)	**0.876 (.11)**
fair	0.909 (.07)	0.751 (.12)	0.637 (.13)	0.547 (.11)	0.731 (.13)	0.637 (.14)	0.635 (.14)	0.653 (.13)	0.601 (.12)	0.131 (.16)
median	0.884 (.09)	0.853 (.10)	0.818 (.12)	0.606 (.17)	0.885 (.09)	0.883 (.09)	0.864 (.10)	0.856 (.09)	0.829 (.11)	0.725 (.15)
Krum	0.838 (.13)	0.864 (.11)	0.818 (.13)	0.768 (.15)	0.847 (.12)	0.870 (.11)	0.805 (.13)	0.868 (.11)	0.866 (.11)	0.640 (.18)
multi-Krum	0.911 (.08)	0.907 (.08)	0.889 (.10)	0.793 (.12)	0.849 (.10)	0.827 (.12)	0.095 (.12)	0.804 (.11)	0.860 (.09)	0.823 (.13)
clipping	0.913 (.07)	0.905 (.08)	0.875 (.10)	0.753 (.12)	0.904 (.08)	0.886 (.09)	0.856 (.11)	0.901 (.08)	0.844 (.11)	0.477 (.13)
k-norm	0.911 (.08)	0.908 (.08)	0.888 (.10)	0.118 (.08)	0.906 (.08)	0.893 (.09)	0.096 (.07)	0.765 (.14)	0.854 (.10)	0.828 (.12)
k-loss	0.898 (.08)	0.856 (.09)	0.861 (.10)	0.851 (.31)	0.876 (.09)	0.866 (.11)	0.870 (.10)	0.538 (.14)	0.257 (.13)	0.092 (.13)
FedMGDA+	0.915 (.08)	0.907 (.08)	0.874 (.10)	0.753 (.13)	0.911 (.08)	0.900 (.09)	0.873 (.10)	**0.914 (.08)**	**0.904 (.08)**	0.869 (.10)
finetuning	0.945 (.06)	**0.946 (.07)**	**0.935 (.07)**	0.922 (.08)	**0.945 (.07)**	**0.930 (.08)**	**0.923 (.08)**	**0.915 (.08)**	0.871 (.11)	0.764 (.15)
`Ditto`, λ=0.1	0.929 (.09)	0.920 (.09)	0.909 (.10)	0.897 (.10)	0.921 (.09)	0.914 (.09)	0.905 (.08)	**0.914 (.09)**	**0.903 (.09)**	**0.873 (.09)**
`Ditto`, λ=1	**0.946 (.06)**	**0.944 (.08)**	**0.935 (.07)**	**0.925 (.07)**	**0.943 (.08)**	**0.930 (.07)**	0.912 (.08)	0.887 (.09)	0.831 (.10)	0.740 (.12)
`Ditto`, λ=2	0.945 (.06)	0.942 (.06)	**0.935 (.07)**	0.917 (.07)	0.936 (.07)	0.923 (.08)	0.906 (.08)	0.871 (.09)	0.785 (.11)	0.606 (.14)

Table 5.10: Full results (average and standard deviation of test accuracy across all devices) on FEMNIST (skewed).

FMNIST (skewed)		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.720 (.24)	0.657 (.28)	0.585 (.30)	0.435 (.23)	0.688 (.26)	0.631 (.24)	0.589 (.26)	0.023 (.11)	0.038 (.18)	0.039 (.18)
local	0.915 (.18)	0.903 (.21)	0.937 (.18)	0.902 (.19)	0.903 (.21)	0.937 (.18)	0.902 (.19)	0.881 (.21)	**0.912 (.18)**	**0.903 (.21)**
fair	0.716 (.22)	0.644 (.29)	0.545 (.29)	0.421 (.22)	0.348 (.22)	0.321 (.16)	0.242 (.15)	0.010 (.11)	0.042 (.10)	0.037 (.17)
median	0.079 (.12)	0.086 (.12)	0.031 (.06)	0.044 (.08)	0.075 (.12)	0.109 (.13)	0.323 (.25)	0.060 (.10)	0.020 (.09)	0.033 (.07)
Krum	0.457 (.37)	0.360 (.35)	0.061 (.22)	0.127 (.27)	0.424 (.38)	0.051 (.08)	0.147 (.22)	0.434 (.36)	0.472 (.36)	0.484 (.35)
multi-Krum	0.725 (.25)	0.699 (.29)	0.061 (.22)	0.271 (.21)	0.712 (.29)	0.705 (.30)	0.584 (.28)	0.633 (.30)	0.556 (.30)	0.526 (.28)
clipping	0.727 (.28)	0.678 (.28)	0.604 (.34)	0.401 (.26)	0.726 (.26)	0.711 (.26)	0.645 (.24)	0.699 (.29)	0.674 (.28)	0.640 (.28)
k-norm	0.716 (.28)	0.691 (.30)	0.396 (.36)	0.005 (.08)	0.724 (.26)	0.721 (.29)	0.692 (.35)	0.612 (.29)	0.599 (.30)	0.565 (.28)
k-loss	0.587 (.21)	0.526 (.29)	0.419 (.36)	0.127 (.27)	0.555 (.23)	0.550 (.26)	0.093 (.16)	0.003 (.08)	0.009 (.07)	0.006 (.05)
finetuning	**0.948 (.11)**	0.942 (.13)	**0.959 (.10)**	**0.946 (.10)**	**0.949 (.16)**	0.918 (.21)	0.621 (.11)	0.788 (.25)	0.740 (.27)	0.751 (.26)
`Ditto`, λ=0.01	0.947 (.15)	**0.945** (.18)	0.955 (.20)	**0.946 (.13)**	0.942 (.18)	0.949 (.15)	**0.944 (.14)**	0.902 (.20)	0.895 (.23)	0.888 (.20)
`Ditto`, λ=0.1	**0.948 (.10)**	**0.945 (.14)**	**0.959 (.12)**	0.936 (.09)	**0.945 (.13)**	**0.948 (.10)**	0.888 (.18)	**0.936 (.16)**	0.827 (.23)	0.812 (.24)
`Ditto`, λ=1	0.902 (.15)	0.899 (.15)	0.907 (.15)	0.861 (.14)	0.899 (.18)	0.818 (.22)	0.423 (.41)	0.880 (.15)	0.730 (.28)	0.736 (.28)

Table 5.11: Full results (average and standard deviation of test accuracy across all devices) on CelebA.

CelebA		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.911 (.19)	0.810 (.22)	0.535 (.26)	0.228 (.21)	0.869 (.22)	0.823 (.23)	0.656 (.26)	0.451 (.27)	0.460 (.29)	0.515 (.31)
local	0.692 (.27)	0.690 (.27)	0.682 (.27)	0.681 (.26)	0.690 (.27)	0.682 (.27)	0.681 (.26)	0.692 (.27)	0.693 (.27)	0.690 (.27)
fair	0.905 (.17)	0.724 (.27)	0.509 (.27)	0.195 (.21)	0.790 (.26)	0.646 (.27)	0.646 (.27)	0.442 (.27)	0.426 (.28)	0.453 (.28)
median	0.910 (.18)	0.872 (.22)	0.494 (.28)	0.126 (.18)	0.901 (.20)	0.864 (.20)	0.617 (.30)	0.885 (.20)	**0.891 (.19)**	**0.870 (.21)**
Krum	0.775 (.25)	0.810 (.25)	0.641 (.25)	0.377 (.10)	0.790 (.25)	0.699 (.25)	0.584 (.27)	0.780 (.25)	0.728 (.25)	0.685 (.30)
multi-Krum	0.911 (.19)	**0.882 (.22)**	0.564 (.26)	0.107 (.19)	0.887 (.21)	0.891 (.20)	0.617 (.30)	0.512 (.27)	0.529 (.27)	0.430 (.26)
clipping	0.909 (.18)	0.866 (.19)	0.485 (.29)	0.126 (.20)	0.897 (.20)	0.842 (.21)	0.665 (.26)	**0.901 (.20)**	0.883 (.21)	0.853 (.23)
k-norm	0.908 (.18)	0.870 (.22)	0.537 (.28)	0.105 (.17)	0.874 (.23)	**0.909 (.18)**	0.664 (.25)	0.506 (.28)	0.577 (.27)	0.449 (.28)
k-loss	0.873 (.19)	0.584 (.28)	0.550 (.31)	0.169 (.21)	0.595 (.28)	0.654 (.28)	0.683 (.26)	0.543 (.33)	0.458 (.33)	0.455 (.34)
FedMGDA+	0.909 (.19)	0.853 (.21)	0.508 (.28)	0.473 (.34)	**0.907 (.19)**	0.889 (.21)	**0.782 (.26)**	0.865 (.23)	0.805 (.26)	0.847 (.21)
finetuning	0.912 (.18)	0.814 (.24)	0.721 (.28)	0.691 (.29)	0.850 (.24)	0.800 (.25)	0.747 (.24)	0.665 (.28)	0.668 (.27)	0.673 (.28)
`Ditto`, λ=0.1	0.884 (.24)	0.716 (.27)	**0.721 (.27)**	**0.724 (.28)**	0.727 (.26)	0.708 (.28)	0.706 (.28)	0.699 (.28)	0.694 (.27)	0.689 (.28)
`Ditto`, λ=1	0.911 (.16)	0.820 (.26)	0.714 (.28)	0.675 (.29)	0.872 (.22)	0.826 (.26)	0.708 (.29)	0.629 (.29)	0.667 (.28)	0.685 (.28)
`Ditto`, λ=2	**0.914 (.18)**	0.828 (.22)	0.698 (.27)	0.654 (.28)	0.862 (.21)	0.791 (.26)	0.623 (.31)	0.585 (.29)	0.647 (.27)	0.655 (.29)

Table 5.12: Full results (average and standard deviation of test accuracy across all devices) on StackOverflow.

StackOverflow		**A1** (ratio of adversaries)			**A2** (ratio of adversaries)			**A3** (ratio of adversaries)		
Methods	clean	20%	50%	80%	20%	50%	80%	10%	15%	20%
global	0.155 (.13)	0.153 (.13)	0.156 (.16)	0.169 (.18)	0.147 (.12)	0.009 (.03)	0.013 (.01)	0.000 (.00)	0.000 (.00)	0.000 (.00)
local	0.311 (.15)	0.311 (.15)	0.313 (.15)	**0.319 (.15)**	0.311 (.15)	0.313 (.15)	**0.319 (.15)**	0.311 (.15)	0.313 (.15)	0.319 (.15)
fair	0.154 (.13)	0.155 (.14)	0.153 (.13)	0.141 (.10)	0.000 (.00)	0.000 (.00)	0.000 (.00)	0.148 (.12)	0.152 (.13)	0.167 (.11)
median	0.002 (.00)	0.001 (.00)	0.000 (.00)	0.000 (.00)	0.000 (.00)	0.001 (.00)	0.000 (.00)	0.000 (.00)	0.000 (.00)	0.000 (.00)
Krum	0.154 (.13)	0.150 (.13)	0.041 (.04)	0.002 (.00)	0.158 (.13)	0.151 (.13)	0.167 (.12)	0.153 (.13)	0.154 (.14)	0.138 (.15)
clipping	0.154 (.13)	0.157 (.13)	0.149 (.13)	0.163 (.17)	0.152 (.13)	0.001 (.01)	0.001 (.01)	0.155 (.12)	0.161 (.14)	0.120 (.16)
k-norm	0.154 (.13)	0.156 (.12)	0.100 (.08)	0.002 (.00)	0.086 (.11)	0.042 (.03)	0.001 (.00)	0.149 (.15)	0.144 (.15)	0.155 (.13)
k-loss	0.155 (.13)	0.160 (.12)	0.164 (.13)	0.129 (.14)	0.136 (.11)	0.145 (.11)	0.156 (.14)	0.148 (.14)	0.159 (.13)	0.156 (.13)
FedMGDA+	0.155 (.12)	0.154 (.13)	0.152 (.13)	0.165 (.13)	0.147 (.13)	0.160 (.14)	0.101 (.09)	0.155 (.13)	0.158 (.12)	0.154 (.13)
`Ditto`, λ=0.05	**0.315 (.16)**	**0.325 (.16)**	**0.315 (.17)**	0.313 (.15)	**0.314 (.16)**	**0.350 (.16)**	0.312 (.14)	0.316 (.17)	**0.321 (.17)**	**0.327 (.17)**
`Ditto`, λ=0.1	0.309 (.17)	0.318 (.17)	**0.315 (.17)**	0.293 (.13)	0.309 (.17)	0.316 (.16)	0.307 (.14)	**0.319 (.17)**	0.302 (.17)	0.305 (.17)
`Ditto`, λ=0.3	0.255 (.18)	0.298 (.18)	0.288 (.17)	0.304 (.16)	0.283 (.17)	0.233 (.18)	0.321 (.20)	0.252 (.17)	0.261 (.19)	0.269 (.17)

Chapter 6

Privacy: Differentially Private Adaptive Optimization

As discussed in Chapter 1, federated learning, by minimizing data exposure, is not inherently private or secure. We explore differential privacy (DP) in federated networks in this chapter. We focus on private adaptive optimization, which is fundamental for current language applications. The proposed methods are effective to reduce privacy noise for general ML applications beyond FL.

6.1 Overview

Privacy-sensitive applications in areas such as healthcare and cross-device federated learning have fueled a demand for optimization methods that ensure *differential privacy (DP)* [4, 49, 82, 210]. These methods typically perturb gradients with random noise at each iteration in order to mask the influence of individual examples on the trained model. As the amount of privacy is directly related to the number of training iterations, private applications stand to benefit from optimizers that improve convergence speed. To capitalize on this, a number of recent works have naturally tried to combine DP with adaptive optimizers such as Adagrad, RMSProp, and Adam, which have proven to be effective for non-private machine learning tasks, especially those involving sparse gradients or non-uniform stochastic noise [78, 117, 155, 237, 318, 321].

Unfortunately, tasks where adaptive optimizers work particularly well (e.g., sparse, high-dimensional problems), are exactly the tasks where DP is known to degrade performance [25]. Indeed, as we show in Figure 6.1, this can result in existing private adaptive optimization methods performing *only marginally better* than simple baselines such as differentially private stochastic gradient descent (DP-SGD), even under generous privacy budgets. The rationale behind this is that to guarantee privacy, estimating the required statistics on noisy gradients can introduce significant noise (Figure 6.2), making these methods less effective.

In this chapter, we aim to close the gap between adaptive optimization in non-private and private settings.

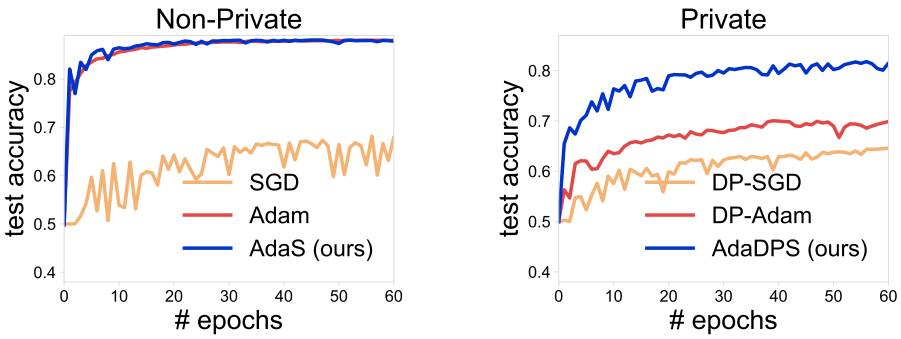


Figure 6.1: Test performance on IMDB with logistic regression. `AdaS` refers to using preconditioning in `AdaDPS` for non-private training. Adaptive methods (Adam) become less effective when trained with differential privacy (DP-Adam), while `AdaDPS` retains the benefits of adaptivity.

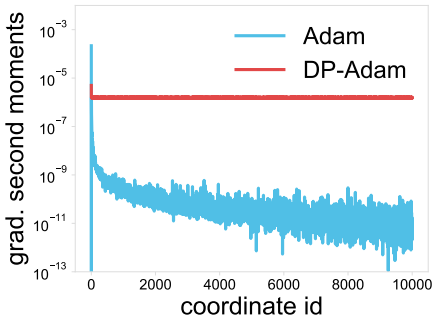


Figure 6.2: The estimates of gradient statistics (e.g., second moments) in private adaptive methods (e.g., DP-Adam) are noisy and may become uninformative of the relative importance of coordinates.

6.1.1 With Side Information

We propose `AdaDPS`, a simple yet powerful framework leveraging non-sensitive side information to effectively adapt to the gradient geometry. Our framework makes two key changes over prior art in private adaptive optimization: (1) Rather than privatizing the gradients first and then applying preconditioners, we show that transforming the gradients prior to privatization can reduce detrimental impacts of noise; (2) To perform gradient transformations, we explore using simple, easily obtainable side information in the data. We discuss two practical scenarios for obtaining such information below.

Public Data as Side Information. A natural choice for side information is to use a small amount of public data generated from a similar distribution as the private data, a common assumption in private optimization [15, 20, 141, 332, 333]. In practice, public data could be obtained through proxy data or from 'opt-out' users who are willing to share their information [12, 141]. Indeed, the notion of heterogeneous DP where subsets of samples require zero or weak privacy has been extensively studied in prior works [e.g.,

11, 137]. Another line of works is to assume that the gradients are low rank, and then use public data to estimate this gradient subspace—thereby mitigating some of the earlier discussed poor performance of DP in high-dimensional regimes. We do not consider using public data for this purpose in this work, as such a low-rank assumption might not hold in practice, particularly for the problems settings where adaptive optimizers are known to excel [20]. Instead, we propose to use the public data more directly: we estimate gradient statistics on public data at each iteration, and then apply these statistics as a preconditioner *before* privatizing the gradients. Despite the simplicity of this procedure, we unaware of any work that has explored it previously.

Summary Statistics as Side Information. Of course, there may also be applications where it is difficult to obtain public data, particularly data that follows the same distribution as the private data. In such scenarios, our insight is that for many applications, in lieu of public data we may have access to some common knowledge about the training data that can be (i) computed before training, and (ii) used to improve optimization performance in both private and non-private settings. For instance, in many language tasks, certain aggregate statistics (e.g., frequency) of different words/tokens are common knowledge or may be easily computed prior to training, and serve as reasonably good estimates of the predictiveness of each feature. `AdaDPS` considers leveraging such simple heuristics to precondition the gradients. Perhaps surprisingly, in our experiments (Section 6.4), we demonstrate that the performance of `AdaDPS` when scaling gradients via these simple statistics (such as feature frequency) can even match the performance of `AdaDPS` with public data.

6.1.2 Without Side Information

Prior works typically address this issue by using non-sensitive auxiliary data or side/meta information of the dataset to approximate the underlying structures of private gradients [20, 142, 187]. While this can boost performance, assuming access to informative public data may be unrealistic in many privacy-sensitive applications. In this section, we instead ask: **Can we improve privacy/utility tradeoffs in private adaptive optimization *without* accessing auxiliary data or side information**?

A key insight we have in addressing this question is that for many machine learning problems, the gradient geometry may not change drastically during successive steps of optimization (e.g., see Figure 6.3, which plots successive distributions of preconditioner values). This presents an opportunity to estimate the preconditioners used by adaptive optimizers with smaller noise, by averaging across previous iterates. To this end, we propose `DP`^{2}, a differentially private adaptive method that uses historical gradients to construct delayed preconditioners with reduced noise. Despite

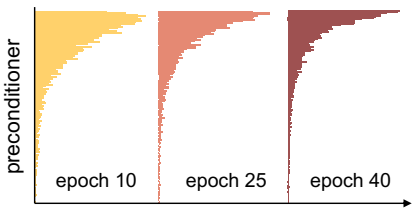


Figure 6.3: Preconditioner values do not change drastically during optimization (IMDB dataset).

the simplicity of this approach, we find that it can significantly improve performance in practice—improving convergence speed by as much as $4\times$ relative to non-adaptive baselines, all without the need to access auxiliary data. To better understand these performance gains, we theoretically and empirically analyze the method to study the effect of using delayed preconditioners, including tradeoffs that emerge between the noise reduction and staleness.

Contributions. We summarize our contributions of this chapter as follows.

- We propose effective optimization frameworks (`AdaDPS` and `DP`^{2}) to approximate the preconditioners in private adaptive optimization. Our algorithms either rely on non-sensitive side information or stale gradients without access to auxiliary information.
- We analyze `AdaDPS` and `DP`^{2} and provide convergence guarantees for both convex and non-convex objectives. In convex cases, we analyze a specific form using RMSProp updates to provably demonstrate the benefits of our approaches relative to differentially private SGD when the gradients are sparse.
- Empirically, we evaluate our method on a set of real-world datasets. They improve the absolute accuracy significantly compared with strong baselines under the same privacy budget, and can even achieve similar accuracy as adaptive methods in non-private settings. We additionally demonstrate how to apply `AdaDPS` to the application of federated learning, where it outperforms existing baselines by a large margin.

Notation. Throughout this chapter, we consider using adaptive optimization methods to solve the classic empirical risk minimization objective, i.e., $\min_w F(w) = \frac{1}{n}\sum_{i=1}^n f(x^i; w)$, where $w \in \mathbb{R}^d$ and $\{f(x^i; w)\}_{i\in[n]}$ are individual loss functions on training sample $i \in [n]$. Optionally, there may exist pubic data denoted as x_{pub}, which does not overlap with $\{x^i\}_{i\in[n]}$. For vectors $u, v \in \mathbb{R}^d$, we use $u + v$ for coordinate-wise addition, and $\frac{u}{v}$ for coordinate-wise division. For any vector v, v_j denotes the j-th coordinate of v. For example, $g_j^{i,t}$ refers to the j-th coordinate of gradient $g^{i,t}$. Finally, $|v| \in \mathbb{R}^d$ denotes taking coordinate-wise absolute values, and $\|\cdot\|_M$ denotes the matrix norm defined as $\|\cdot\|_M := \sqrt{\langle \cdot, M\cdot\rangle}$ for a symmetric and positive definite matrix $M \in \mathbb{R}^{d\times d}$, or a diagonal matrix with non-negative diagonal entries populated by a vector $M \in \mathbb{R}^d$.

In terms of privacy formulations, we consider classic sample-level DP in centralized settings, and a variant of it—*user-level DP*—in distributed/federated environments. They are formally defined in Chapter 2. We focus primarily on the classic centralized training, and later on extend it to federated settings (Objective (6.1)) [206].

6.2 `AdaDPS`: Private Adaptive Optimization With Side Information

Gradient-based private optimization methods usually update the model parameters with noisy gradients at each iteration and then release the private final models [4]. To control

the sensitivity of computing and summing individual gradients from a mini-batch, methods based on the subsampled Gaussian mechanism typically first clip each individual gradient and then add i.i.d. zero-mean Gaussian noise with variance determined by the clipping threshold and the privacy budget. To use adaptivity effectively, in AdaDPS, we instead propose first preconditioning the raw gradients with side information estimated either on public data or via some auxiliary knowledge, and then applying the Gaussian mechanism with noise multiplier σ on top. AdaDPS in centralized training is summarized in Algorithm 8. Note that AdaDPS is a general framework in that it incorporates a set of private adaptive methods. As described in Algorithm 8, the functions ϕ, φ, and $\mathcal{A}$ abstract a set of updating rules of different adaptive methods. Next, we describe the algorithm in more detail and instantiate ϕ, φ, and $\mathcal{A}$.

Option 1 (With Public Data). We first consider estimating gradient statistics based on public data. Functions ϕ, φ, and $\mathcal{A}$ can define a very broad a set of common adaptive updating rules, as shown below.

- *Adam*: A^t is the square root of the second moment estimation, and M^t is the first moment estimation; with $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{\beta^t g^{i,t} + (1-\beta^t) M^t}{A^t}$ for some moving averaging parameter β^t.
- *AdaGrad*: $M^t = \mathbf{0}$, $(A^t)^2 = (A^{t-1})^2 + (\hat{g}^t)^2$, and $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{g^{i,t}}{A^t}$.
- *RMSProp*: A^t is the square root of the second moment estimation with $M^t = \mathbf{0}$. And $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{g^{i,t}}{A^t}$.

We note that the AdaDPS framework can potentially incorporate other adaptive optimizers, beyond what is listed above. In our analysis and experiments, we mainly focus on using RMSProp updates to obtain the preconditioner, because AdaGrad which sums up gradients in all iterations so far in the denominator, often has poor practical performance, and Adam needs to maintain an additional mean estimator. However, we also evaluate the use of Adam within AdaDPS in Table 6.6 in Appendix 6.7, showing that it yields similar improvements as RMSProp across all datasets. In Section 6.3, we analyze the convergence of A^t as $\mathbb{E}[(\hat{g}^t)^2]$, and prove that in sparse settings, AdaDPS allows the addition of less noise under the same privacy budget.

Option 2 (Without Public Data). When there is no public data available, we develop simple and effective heuristics to determine which coordinates are more predictive based on non-sensitive side information. In particular, for generalized linear models in NLP applications, we set A^t to be (i) proportional to the frequency of input tokens, or (ii) proportional to the TF-IDF values of input tokens. Follow a similar analysis as that of Option 1, we provide theoretical justification in Theorem 17 in Section 6.3.1.1. While these are simple approaches to remove the dependence on public data, we find that they can significantly outperform DP-SGD for real-world tasks with several million model parameters (Section 6.4.1).

Algorithm 8 `AdaDPS`

Input: T, batch size b, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, side information $A^t \in \mathbb{R}^d$, learning rate α^t, potential momentum buffer $M^0 \in \mathbb{R}^d$

for $t = 1, \cdots, T-1$ **do**

Uniformly sample a mini-batch B ($|B|$=b) from the training set and get b gradients:

$$g^{i,t} \leftarrow \nabla f(x^i; w^t),\ i \in B$$

Option 1: With public data x_{pub}

Uniformly sample a mini-batch B' ($|B'|$=b) from x_{pub}, get gradients, and update A^t and M^t with recurrence ϕ and φ respectively:

$$\hat{g}^t \leftarrow \frac{1}{b} \sum_{j \in B'} \nabla f(x^j; w^t),\ x^j \in x_{\text{pub}}$$
$$A^t \leftarrow \phi(A^{t-1}, \hat{g}^t), M^t \leftarrow \varphi(M^{t-1}, \hat{g}^t),$$

Option 2: Without public data

$$A^t \text{ estimated via heuristics}$$

Precondition individual gradients by $\mathcal{A}$:

$$g^{i,t} \leftarrow \mathcal{A}(g^{i,t}, A^t, M^t)$$

Privatize preconditioned gradients:

$$\tilde{g}^t \leftarrow \frac{1}{b} \sum_{i \in B} \text{clip}\left(g^{i,t}, C\right) + \frac{1}{b} \mathcal{N}\left(0, \sigma^2 C^2\right),$$

where $\text{clip}(g, C)$ clips a vector g to L_2 norm C

Update the model parameter w as

$$w^{t+1} \leftarrow w^t - \alpha^t \tilde{g}^t$$

end for

return w^T

Privacy Guarantees. We now state the differential privacy guarantees of Algorithm 8. As the side information A^t (as well as the potential momentum buffer M^t) is non-sensitive, its privacy property directly follows from previous results [4].

Theorem 15 (Privacy guarantee of Algorithm 8 [4])**.** *Assume the side information A^t is*

non-sensitive. There exist constants c_1 and c_2 such that for any $\varepsilon < c_1 b^2 T/n^2$, Algorithm 8 is (ε, δ)-differentially private for any $\delta > 0$ if $\sigma \geqslant c_2 \frac{b\sqrt{T\log(1/\delta)}}{n\varepsilon}$.

6.2.1 Intuition for A^t

In this section we provide further intuition for the `AdaDPS` framework. When A^t is an all-ones vector, `AdaDPS` reduces to the normal DP-SGD algorithm. Otherwise, A^t is indicative of how informative each coordinate is. Intuitively, suppose clipping does not happen and the public data come from the same distribution as private data so that for the RMSProp preconditioner, we have $A^t = \sqrt{\mathbb{E}[(g^{i,t})^2]}$. Then the effective transformation on each individual gradient $g^{i,t}$ is $\frac{g^{i,t} + \mathcal{N}\left(0, \sigma^2 C^2 \mathbb{E}[(g^{i,t})^2]\right)}{\sqrt{\mathbb{E}[(g^{i,t})^2]}}$. This can be viewed as first adding non-isotropic noise proportional to the second moment of gradients, and then applying RMSProp updates, which is beneficial as coordinates with higher second moments are more tolerant to noise. Therefore, `AdaDPS` could improve privacy/utility tradeoffs via adding coordinate-specific noise (formalized in Theorem 16).

We next consider a toy example to highlight one of the regimes where `AdaDPS` (or, adaptive methods) is particularly effective. Consider a linear regression task with the objective $\min_{w \in \mathbb{R}^{500}} \frac{1}{2n} \sum_{i \in [n]} (w^\top x^i - y^i)^2$ where $n = 1,000$ and each sample $x^i \in \mathbb{R}^{500}$. In many real-world applications, the tokens (features) are sparse and their frequencies follow heavy-tailed distributions. Without loss of generality, we assume the first 10% features are frequent and uninformative; and the later 90% rare and informative. Let the j-th feature of all data points be sampled from a random variable $x_j \in \{0, 1\}$. Features and the underlying true w are generated as follows:

$$\Pr(x_j = 1) = \begin{cases} 0.9, & j \leqslant 50 \\ 0.01, & j > 50 \end{cases}, \quad w_j = \begin{cases} 0.01, & j \leqslant 50 \\ 1.0, & j > 50 \end{cases}.$$

Labels are generated by $y^i = \langle w, x^i \rangle + b^i$ where $b^i \sim \mathcal{N}(0, 0.01)$. For `AdaDPS`, we assume model engineers know which words are more frequent, thus setting $A_j^t = 1$ for $j \leqslant 50$ and $A_j^t = 0.01$ otherwise for all t. Using larger learning rates on informative coordinates, side information helps to improve optimization performance dramatically (see results in Figure 6.4).

Comparison to Asi et al. [20]. The most related work to ours is Asi et al. [20], which adds non-isotropic noise which lies in a non-uniform ellipsoid to the gradients, and (optionally) transforms the resulting gradients with the demoninator used in AdaGrad. `AdaDPS` differs from their approach in several ways, as we (i) first precondition then add noise (as opposed to the other way round), (ii) consider a broader class of preconditioners (beyond AdaGrad), and (iii) make the approaches to estimating gradient geometry in Asi et al. [20] more explicit in lieu of public data (as discussed in previous sections). Empirically, we compare with another state-of-the-art method [15] which outperforms Asi et al.

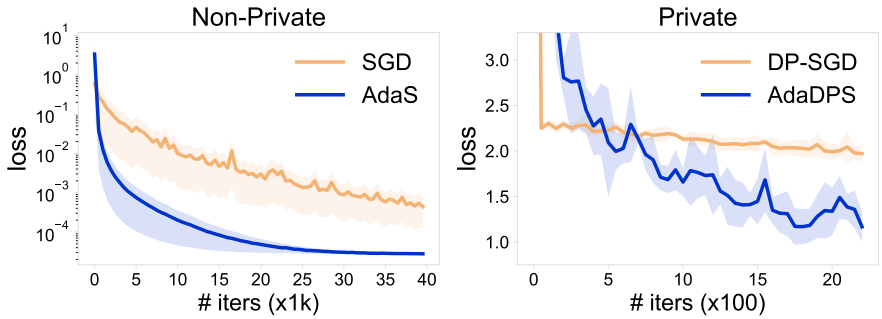


Figure 6.4: Training loss on the linear regression problem described in Section 6.2 (averaged over five runs). We tune optimal learning rates separately for each method. Private training (right) achieves $(4.13, 10^{-3})$-DP.

[20], and demonstrate AdaDPS's superior performance (Section 6.4, Table 6.2).[1]

6.3 Convergence Analysis of AdaDPS

We now analyze the convergence of AdaDPS (Algorithm 8) with and without public data, in both convex (Section 6.3.1) and non-convex (Section 6.3.2) settings. When there is public data available, we prove that AdaDPS adds less noise (plus, with noise proportional to the magnitude of gradients) compared with DP-SGD. Our theory extends previous proofs in related contexts, but considers stochastic gradients, adding random Gaussian noise to the processed gradients, and estimating the preconditioner on public data (as opposed to updating it with the raw gradients on training data). When there is no public data, we present convergence results for general A^t, covering the heuristics used in practice.

6.3.1 Convex Convergence

For convex functions, we define the optimal model w^* as $w^* \in \arg\min_w F(w)$. First we state a set of assumptions that are used in the analyses.

Assumption 2. *There exists a constant D such that $\|w^t - w^*\|_2 \leqslant D$ for any $t \in [T]$.*

Assumption 3. *There exists a constant C such that $\left\|\frac{g^{i,t}}{A^t}\right\|_2 \leqslant C$ for any $t \in [T]$ and $i \in [n]$.*

Assumption 4. *Denote $g^t := \frac{1}{b}\sum_{i\in B} g^{i,t}$. There exists a constant $a \in (0, 1]$ such that for any $j \in [d]$ and $t \in [T]$, $a(\hat{g}_j^t)^2 \leqslant (g_j^t)^2 \leqslant \frac{1}{a}(\hat{g}_j^t)^2$ holds.*

Assumption 2 (bounded domain across all iterations) is commonly used in adaptive optimization literature [20, 168, 187, 237]. Assumption 3 aims to bound the L_2 norm

[1] We do not compare with Asi et al. [20] directly as the code is not publicly available.

of the stochastic gradient, thus helping bound the L_2 sensitivity of the operation of calculating and averaging individual gradients from a mini-batch. Assuming bounded stochastic gradient norm is standard in prior works on convex and non-convex private optimization [e.g., 142, 187, 332]. Assumption 4 bounds the dissimilarity between public and private data.

Within the framework of `AdaDPS`, we explore the convergence of the RMSProp preconditioner where $A^t = \sqrt{\mathbb{E}[(\hat{g}^t)^2]} + \epsilon^t$ (with public data) where ϵ^t is some small constant or A^t is obtained via side information heuristics. We first look at the case where there exist public data, and therefore the exact updating rule at the t-th iteration is:

$$\hat{g}^t, g^t \leftarrow \frac{1}{b}\sum_{j\in B'} \nabla f(x^j; w^t)\ (x^j \in x_{\text{pub}}), \frac{1}{b}\sum_{i\in B} \nabla f(x^i; w^t),$$

$$v^t \leftarrow \beta^t v^{t-1} + (1-\beta^t)(\hat{g}^t)^2,\ A^t \leftarrow \sqrt{v^t} + \epsilon^t,$$

$$w^{t+1} \leftarrow w^t - \alpha^t \left(\frac{g^t}{A^t} + N\right),\ N \sim \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2).$$

Theorem 16 below states the convergence guarantees.

Theorem 16. *Assume $F(w)$ is a convex function w.r.t. w. Let Assumptions 2-4 hold. Additionally, choose β^t such that $1 - \frac{\gamma}{t} \geqslant \beta^t \geqslant 1 - \frac{1}{t}$ holds for some $\gamma \in (0, 1]$; and let $\sqrt{t+1}\epsilon^{t+1} \geqslant \sqrt{t}\epsilon^t$ for any t. After running Algorithm 8 using learning rates $\alpha^t = \frac{\alpha}{\sqrt{t}}$ with public data for T iterations, we have*

$$\min_{t\in[T]} \mathbb{E}[F(w^t)] - F^* \leqslant \frac{G}{\sqrt{T}} \sum_{j=1}^{d} \mathbb{E}\left[A_j^T\right] + \frac{\alpha}{\sqrt{T}} \max_{t\in[T]} \mathbb{E}\left[\|N\|^2_{A^t}\right],$$

where $F^* := F(w^*)$, $G = \frac{D^2}{2\alpha} + \frac{\alpha(2-\gamma)}{a\gamma}$, and $A_j^T = \sqrt{v_j^T} + \epsilon^T$.

The full proof is deferred to Appendix 6.5.1. Our proof extend the proof of the original RMSProp method with full gradients in Mukkamala and Hein [218] to stochastic and private settings. From Theorem 6.5, we see that the first term in the bound is standard for the RMSProp optimizer, and the last term is due to noise added to ensure differential privacy. Fixing the noise multiplier σ, the second term depends on the clipping value C and the preconditioner A^t. We see that when the gradients are sparse, it is likely that the added DP noise would be significantly reduced. In other words, to guarantee overall (ε, δ)-differential privacy by running T total iterations, we can set $\sigma^2 = O\left(\frac{b^2 T \log(1/\delta)}{n^2\varepsilon^2}\right)$ and $T = O\left(\frac{n^2\varepsilon^2}{\max_{t\in[T]} \mathbb{E}[\|A^t\|_1] \log(1/\delta)}\right)$. The convergence rate therefore becomes

$$\min_{t\in[T]} \mathbb{E}[F(w^t)] - F^* \leqslant O\left(\frac{\sqrt{\max_{t\in[T]} \mathbb{E}\left[\|A^t\|_1\right]}}{n\varepsilon}\right).$$

When gradients are sparse (hence $\max_{t\in[T]} \mathbb{E}\left[\left\|A^t\right\|_1\right] < d$), the amount of noise added will be significantly smaller compared with that of vanilla DP-SGD to guarantee the same level of privacy. This highlights one regime where AdaDPS is particularly useful, though AdaDPS also yield improvements in other settings with dense gradients (Table 6.6 in the appendix). Here, Theorem 16 assumes access to $\mathbb{E}[(\hat{g}^t)^2]$. When there is no public data available, we leverage other easily obtainable side information to determine fixed A^t prior to training, as analyzed in the next section.

6.3.1.1 Fixed A^t

Theorem 17. *Let assumptions in Theorem 16 hold. Running Algorithm 8 using learning rates $\alpha^t = \frac{\alpha}{\sqrt{t}}$ without public data with side information $A \in \mathbb{R}^d$ for T iterations gives*

$$\min_{t\in[T]} \mathbb{E}[F(w^t)] - F^* \leqslant O\left(\frac{\alpha R + 1}{\sqrt{T}} \sum_{j=1}^{d} A_j + \frac{\alpha}{\sqrt{T}} \mathbb{E}\left[\|N\|_A^2\right]\right),$$

where $R := \max_{j,t} \frac{\mathbb{E}[(g_j^t)^2]}{A_j^2}$ and $g^t := \frac{1}{b}\sum_{i\in B} g^{i,t}$.

From Theorem 17 above, we observe that A should be chosen such that both R and $\sum_{i=j}^{d} A_j$ are minimized. For a large class of generalized linear models, we are able to obtain appropriate A based on the feature space information to control the values of R, as discussed in the following.

Considering generalized linear models. Under the model parameter $w \in \mathbb{R}^d$, for any x^i, the gradients are $c(x^i; w)x^i$ where $c(x^i; w) \in \mathbb{R}$ is a function of x^i and w. We assume that for any $i \in [n]$ and w, there exists a constant $c_{\max}$ such that $|c(x^i; w)| \leqslant c_{\max}$. One natural choice of A is to set $A_j = \sqrt{\mathbb{E}[x_j^2]} + \epsilon$ for each coordinate $j \in [d]$ (such that $R \leqslant c_{\max}^2$), which could improve the noise term $\mathbb{E}\left[\|N\|_A^2\right]$ when the features are sparse. Nevertheless, $\mathbb{E}[x^2]$ can be unrealistic to obtain prior to training. Instead, one side information of choice is $\mathbb{E}[x]$, which implies how rare the raw features are in some NLP applications. Let $A_j = \mathbb{E}[|x_j|] + \epsilon$ $(j \in [d])$. Then

$$R = \max_{j,t} \frac{\mathbb{E}\left[(g_j^t)^2\right]}{\left(\mathbb{E}[|x_j|] + \epsilon\right)^2} \leqslant \max_{j,t} \frac{c_{\max}^2 \mathbb{E}[x_j^2]}{(\mathbb{E}[|x_j|])^2 + \epsilon^2}.$$

To reason about how large R is, we consider a simple setup where each feature takes the value of $v > 0$ with probability p, and 0 with probability $1 - p$. It is straightforward to see the scaling of the last two terms in the convergence bound in Theorem 17:

$$R \sum_{j=1}^{d} A_j = O\left(\frac{1}{p}\right) O\left(dpv\right) = O(dv),$$

$$\mathbb{E}\left[\|N\|_A^2\right] = \sigma^2 C^2 \sum_{j=1}^{d} A_j = O\left(\max_{i,t} \|g^{i,t}\|^2 \frac{dpv}{(pv+\epsilon)^2}\right).$$

$\mathbb{E}\left[\|N\|_A^2\right]$ will reduce to $O(d)$ if the gradients are sparse in a certain way, i.e., $\max_{i,t} \|g^{i,t}\|^2 = O(p)$. Note that only using this simple first-moment information ($A_j = \mathbb{E}\left[|x_j|\right] + \epsilon$), we are not able to obtain a constant improvement in the convergence bound as in Theorem 16 with public data. However, we empirically demonstrate that these ideas can be effective in practice in Section 6.4.

6.3.2 Non-Convex Convergence

In addition to convex problems, we also consider convergence to a stationary point for non-convex and smooth functions. We introduce additional assumptions in the following.

Assumption 5. *Each $f(x^i; w)$ $(i \in [n])$ is L-smooth with respect to $w \in \mathbb{R}^d$.*

Assumption 6. *The expectation of stochastic gradient variance is bounded, i.e., $\mathbb{E}[\|g_j^{i,t} - \mathbb{E}[g_j^{i,t}]\|_2^2] \leqslant \tau_j^2$ for all i, t, j. Denote $\tau^2 := (\tau_1^2, \cdots, \tau_d^2) \in \mathbb{R}^d$.*

Assumption 5 together with Assumption 2 implies that there exists a constant that bounds $\|\nabla F(w^t)\|$ for any t, which we denote as B.

Theorem 18. *Let Assumptions 2-6 hold. After running Algorithm 8 with public data for T iterations using a constant learning rate α and a constant ϵ, choosing the constants to satisfy $\alpha \leqslant \frac{\epsilon}{2L}$ and $B\sqrt{1-\beta} \leqslant \frac{\sqrt{a}\epsilon}{4}$, we have*

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}\left[\|\nabla F(w^t)\|^2\right] \leqslant O\left(\frac{1}{T}\right) + O\left(\frac{\|\tau^2\|_1}{b} + \frac{\sigma^2}{b^2}\right).$$

The proof is mostly extended from Adam's proof in Zaheer et al. [318] (see Appendix 6.5.2 for complete steps). When the batch size increases, both the stochastic gradient noise and differential privacy noise would be reduced.

Theorem 19. *Let Assumptions 2-6 hold. After running Algorithm 8 with a fixed A as prior information for T iterations using a constant learning rate α and a constant ϵ, choosing the constants to satisfy $\alpha \leqslant \frac{\epsilon}{L}$, we have*

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}\left[\|\nabla F(w^t)\|_{A^{-1}}^2\right] \leqslant O\left(\frac{1}{T}\right) + O\left(\frac{\tau^2}{A}\frac{1}{b} + \frac{\sigma^2}{b^2}\right).$$

6.4 Experiments on AdaDPS

We evaluate the performance of AdaDPS in both centralized (Section 6.4.1) and federated (Section 6.4.2) settings for various tasks and models. In centralized training, we investigate two practical scenarios for obtaining side information with and without public data (Section 6.4.1.1 and 6.4.1.2). We describe our experimental setup below; details of datasets, models, and hyperparameter tuning are described in Appendix 6.6. Our code is publicly available at github.com/litian96/AdaDPS.

Datasets. We consider common benchmarks for adaptive optimization in centralized or federated settings [15, 235, 318] involving varying types of models (both convex and non-convex) and data (both text and image data). Both linear and non-convex models contain millions of learnable parameters.

Hyperparameters. We fix the noise multiplier σ for each task, and select an individual (fixed) clipping threshold for each differentially private run. To track the privacy loss (to ensure (ε, δ)-DP), we add the same amount of noise to all compared methods, set the δ value to be the inverse of the number of all training samples, and compute ε using Rényi differential privacy (RDP) accountant for the subsampled Gaussian mechanism [213].

6.4.1 Centralized Training

Common Baselines. One can directly privatize an adaptive optimizer by *first privatizing* the raw gradients, and *then applying that adaptive method* on top of noisy gradients [332]. We consider these baselines named DP-Adam or DP-RMSProp where the adaptive optimizer is chosen to be Adam or RMSProp (same as DP-Adam appearing in previous sections). As the empirical results of DP-Adam and DP-RMSProp are very similar (Table 6.6 in the appendix), in the main text, we mainly compare AdaDPS with DP-Adam [332] and DP-SGD [4]. For completeness, we present the exact DP-Adam algorithm in Appendix 6.6.

6.4.1.1 With public data

In the main text, we set the public data size to be 1% of training data size. We further explore the effects of public data size empirically in Table 6.10, Appendix 6.7. Next, we present results of comparing AdaDPS with several baselines, and results using both in-distribution (ID) and out-of-distribution (OOD) data as public data to estimate the preconditioner.

Preconditioning *Noisy* Gradients with Public Data. In addition to DP-SGD and DP-Adam mentioned in Section 6.4.1, we consider another method of preconditioning the *noisy* gradients with second moment estimates obtained from *public data*. Specifically, the

updating rule at iteration t is

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i\in B} \text{clip}\left(g^{i,t}, C\right) + \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2),$$

$$\tilde{g}^t \leftarrow \frac{\tilde{g}^t}{\sqrt{\mathbb{E}[(\hat{g}^t)^2]} + \epsilon^t}, \text{ where } \hat{g}^t := \nabla f(x; w^t) \text{ for } x \in x_{\text{pub}}.$$

We call this adaptive baseline DP-R-Pub, which is equivalent to standard DP-RMSProp but using public data to estimate the preconditioner. Comparing with this method directly reflects the importance of the order of preconditioning and privatization in `AdaDPS`.

Results with in-distribution proxy data (randomly sampled from training sets) are shown in Figure 6.5 and Table 6.1 below. We see that across three datasets, (i) DP-Adam does not necessarily outperform DP-SGD all the same, (ii) `AdaDPS` improves over all baselines significantly, including DP-R-Pub. Full results involving DP-RMSProp and `AdaDPS` with Adam as the updating rule are presented in Table 6.6.

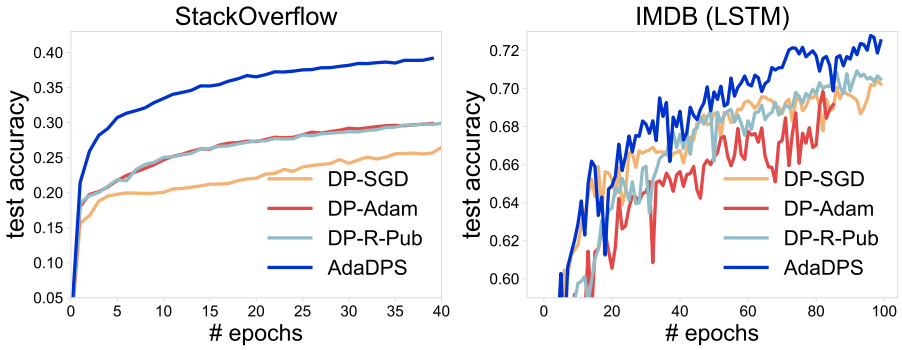


Figure 6.5: Test accuracies of baselines and `AdaDPS` assuming access to public data. ε values on these two datasets are 0.84 and 2.8, respectively. `AdaDPS` significantly improves test performance, even reaching an accuracy much higher than the accuracy of SGD in non-private training on StackOverflow.

Methods	**Loss** $\times 100$ (σ=1)	**Loss** $\times 100$ (σ=0.75)
DP-SGD	5.013 (.001)	3.792 (.001)
DP-Adam	3.697 (.020)	3.286 (.016)
`AdaDPS`	**3.566** (.008)	**3.158** (.003)

Table 6.1: Test reconstruction loss (mean and standard deviation across three runs) on MNIST under a deep autoencoder model. σ=1 and σ=0.75 correspond to ε=1.6 and ε=3, respectively. DP-Adam works well in this task compared with DP-SGD. `AdaDPS` improves over DP-Adam.

We also evaluate `AdaDPS` on MNIST, and observe that it yields $0.5\% - 2\%$ accuracy improvements (Table 6.6), depending on the specific adaptive method.

Comparisons with Amid et al. [15]. We compare AdaDPS with one recent work, PDA-DPMD, which is the state-of-the-art that leverages public data to improve privacy/utility tradeoffs in a mirror descent framework [15]. We take their proposed approximation, where the actual gradients are a convex combination of gradients on private and public data. As this approximation does not precondition gradients, PDA-DPMD could underperform AdaDPS in the tasks where adaptivity is critical, as shown in Table 6.2 below.

Datasets	**Metrics**	**PDA-DPMD** w/ public	AdaDPS w/ public	AdaDPS w/o public
IMDB	accuracy	0.62	**0.80**	0.75
StackOverflow	accuracy	0.33	**0.40**	**0.41**
MNIST	loss	0.039	**0.036**	—

Table 6.2: Comparison with a recent method (PDA-DPMD) using public data in private mirror descent. AdaDPS outperforms PDA-DPMD due to preconditioning.

Out-Of-Distribution Public Data. As mentioned in Section 6.1, public data could be obtained via a small amount of proxy data or 'opt-out' users that do not need privacy protection. We consider two practical cases where we use OOD data to extract side information. For IMDB sentiment analysis, we use a small subset of Amazon reviews[2] [324] as public data (1% the size of IMDB). Amazon reviews study a more fine-grained 5-class classification problem on product reviews, and we map labels {1, 2} to 0 (negative), and labels {4, 5} to 1 (positive). For StackOverflow tag prediction task which consists of 400 users with different styles and interested topics, we simply hold out the first four of them to provide public data. We show results in Table 6.3 below. We see that the preconditioners obtained from out-of-distribution but related public data are fairly effective.

Datasets	**DP-SGD**	AdaDPS OOD public	AdaDPS ID public
IMDB	0.63	**0.79**	**0.80**
StackOverflow	0.28	**0.40**	**0.40**

Table 6.3: Using small out-of-distribution data as public data achieves the same improvements. For IMDB, we leverage Amazon reviews data (1% the size of IMDB) as public data. For StackOverflow, we hold out 1% users as those who opt out of privacy training.

6.4.1.2 Without Public Data

When it is difficult to obtain public data that follow sufficiently similar distribution as private training data, we explore two specific heuristics as side information tailored to

[2] figshare.com/articles/dataset/Amazon_Reviews_Full/13232537/1

language tasks: token frequencies and TF-IDF values of input tokens (or features). These statistics are always known as open knowledge, thus can be used as an approximate how important each feature is. We compare AdaDPS with DP-SGD and DP-Adam.

A^t Based on Token Frequencies. One easily obtainable side information is token frequencies, and we can set A^t ($t \in [T]$) to be proportional to that accordingly. Note that our implicit assumption here is that rare features are more informative than frequent ones. We investigate the logistic regression model on two datasets in Figure 4 below. Despite the simplicity, this simple method works well on StackOverflow and IMDB under a tight privacy budget, outperforming DP-SGD and DP-Adam significantly. Especially for StackOverflow, the test accuracy is the same as that of AdaDPSwith a small set of public data (Figure 6.5).

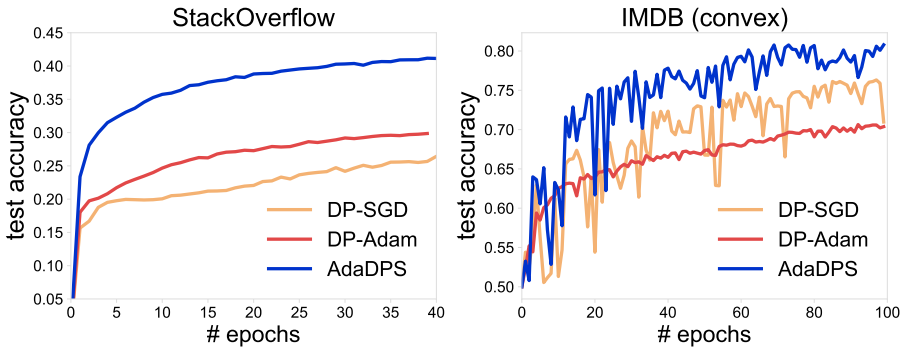


Figure 6.6: AdaDPS uses token frequencies as the side information, demonstrating superior performance than the baselines. Methods in each subfigure reach $(0.84, 4.2 \times 10^{-6})$- and $(2.8, 4 \times 10^{-5})$-DP.

A^t Based on TF-IDF Values. Another common criterion to measure the relevance of tokens to each data point in a collection of training data is TF-IDF values. With the presence of such information available in a data processing pipeline, we explore the effects of A^t being inversely proportional to TF-IDF scores for linear models. The results are reported in Table 6.4. The 'ideal' method refers to AdaDPS with RMSProp updates and the second moment estimated on clean gradients from private training data, which serves a performance upper bound. As expected, across all methods, TF-IDF features result in higher accuracies than BoW features. AdaDPS outperforms the baselines by ~10% absolute accuracy, only slightly underperforming the 'ideal' oracle.

Remark (Side Information in Non-Private Training). The idea of using side information (with or without public data) can also improve the performance of vanilla SGD in non-private training, yielding similar accuracies as that of adaptive methods. We report additional results along this line in Table 6.9 in the appendix.

Features	Methods			
	DP-SGD	DP-Adam	AdaDPS	*ideal*
BoW	0.62 (.02)	0.68 (.01)	**0.75** (.01)	*0.82 (.01)*
TF-IDF	0.68 (.01)	0.65 (.01)	**0.80** (.00)	*0.83 (.00)*

Table 6.4: We preprocess IMDB into two versions with either BoW or TF-IDF features, and report average test accuracy along with standard deviation across three runs. AdaDPS with A^t being inversely proportional to features' TF-IDF values outperforms the baselines of DP-SGD and DP-Adam by a large margin. AdaDPS also performs relatively closely to the 'ideal' upper bound.

6.4.2 Applying AdaDPS to Federated Learning

In this section, we discuss AdaDPS adapted to FL to satisfy user-level, global differential privacy (assuming a trusted central server) (Definition 4). We consider the default objective (1.1) for FL of fitting a model $w \in \mathbb{R}^d$ to data across a network of N devices, i.e.,

$$\min_{w \in \mathbb{R}^d} F(w) = \sum_{k=1}^{N} p_k f_k(w), \tag{6.1}$$

where $f_k(w)$ is the empirical local loss on each device k, and p_k is some pre-defined weight for device k such that $\sum_{k=1}^{N} p_k = 1$. In this work, we simply set $p_k = \frac{1}{N}$, $k \in [N]$. For this privacy-sensitive application, we assume there is no public data available.

Due to the practical constraints of federated settings (e.g., unreliable network conditions, device heterogeneity, etc), federated optimization algorithms typically randomly samples a small subset of devices at each communication round, and allows each device to run optimization methods locally (e.g., local mini-batch SGD) before sending the updates back to the server [206]. Adapting AdaDPS to federated learning is not straightforward, raising questions of applying preconditioning at the server side, the device side, or both [290]. We empirically find that on the considered dataset, preconditioning the mini-batch gradients *locally at each iteration* demonstrates superior performance than preconditioning the entire model updates at the server side. The exact algorithm is summarized in Algorithm 10 in the appendix.

We investigate the same StackOverflow dataset described in Section 6.4.1, but follow its original, natural partition (by Stack Overflow users) for federated settings, one device per user. There are 400 devices in total for the subsampled version we use. We select 20 devices at each communication round and use a noise multiplier $\sigma = 0.3$. While we arrive at a large ε value for user-level DP, the final model could still be useful for defending against membership inference attacks in practice [143].

In Figure 6.7, we plot test accuracy versus the number of communication rounds. AdaDPS has $\sim$5% higher test accuracy than other two methods. We note that in federated learning applications involving massive and unreliable networks, it is not always realistic to allow for uniform device sampling. Incorporating recent advances in DP without sampling [e.g., 143] to address this is left for future work.

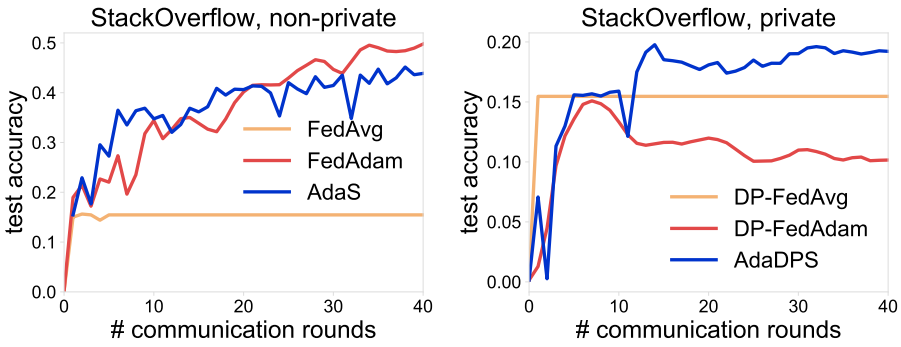


Figure 6.7: Test accuracy of StackOverflow in non-private and private settings under $(34, 0.0025)$ user-level DP (Definition 4). `AdaDPS` extended to federated learning (Algorithm 10 in the appendix) improves over baselines of DP-FedAvg [210] and DP-FedAdam by 5% in terms of test accuracy.

6.5 Convergence Proofs

6.5.1 Proof for Theorem 16 (Convex cases)

Based on our assumptions, the updating rule becomes

$$g^t \leftarrow \frac{1}{b}\sum_{i\in B}\nabla f(x^i; w^t) \tag{6.2}$$

$$\hat{g}^t \leftarrow \frac{1}{b}\sum_{j\in B'}\nabla f(x^j; w^t),\ x^j \in x_{\text{pub}} \tag{6.3}$$

$$v^t \leftarrow \beta^t v^{t-1} + (1-\beta^t)(\hat{g}^t)^2 \tag{6.4}$$

$$A^t \leftarrow \sqrt{v^t} + \epsilon^t \tag{6.5}$$

$$w^{t+1} \leftarrow w^t - \alpha^t\left(\frac{g^t}{A^t} + N\right),\ N \sim \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2) \tag{6.6}$$

We extend the proof in Mukkamala and Hein [218] to stochastic, private cases with preconditioner estimated on public data. Based on the updating rule, we have

$$\left\|w^{t+1} - w^*\right\|_{A^t}^2 \tag{6.7}$$

$$= \left\|w^t - \alpha^t (A^t)^{-1} g^t - \alpha^t N - w^*\right\|_{A^t}^2 \tag{6.8}$$

$$= \left\|w^t - w^*\right\|_{A^t}^2 + \left\|\alpha^t (A^t)^{-1} g^t + \alpha^t N\right\|_{A^t}^2 - 2\left\langle w^t - w^*, \alpha^t g^t + \alpha^t A^t N\right\rangle \tag{6.9}$$

$$= \left\|w^t - w^*\right\|_{A^t}^2 - 2\alpha^t\left\langle g^t, w^t - w^*\right\rangle + (\alpha^t)^2\left\langle g^t, (A^t)^{-1} g^t\right\rangle - 2\alpha^t\langle w^t - w^*, A^t N\rangle$$
$$+ (\alpha^t)^2\|N\|_{A^t}^2 + 2(\alpha^t)^2\langle g^t, N\rangle. \tag{6.10}$$

Rearranging terms gives

$$\left\langle g^t, w^t - w^* \right\rangle = \frac{\|w^t - w^*\|_{A^t}^2 - \|w^{t+1} - w^*\|_{A^t}^2}{2\alpha^t} + \frac{\alpha^t}{2}\left\langle g^t, (A^t)^{-1} g^t \right\rangle - \langle w^t - w^*, A^t N\rangle + \frac{\alpha^t}{2}\|N\|_{A^t}^2 + \alpha^t \langle g^t, N\rangle. \tag{6.11}$$

Take expectation on both sides conditioned on w^t,

$$\left\langle \nabla F(w^t), w^t - w^* \right\rangle = \frac{\mathbb{E}_t\left[\|w^t - w^*\|_{A^t}^2\right] - \mathbb{E}_t[\|w^{t+1} - w^*\|_{A^t}^2]}{2\alpha^t} + \frac{\alpha^t}{2}\mathbb{E}_t\left[\left\langle g^t, (A^t)^{-1} g^t \right\rangle\right] + \frac{\alpha^t}{2}\mathbb{E}_t\left[\|N\|_{A^t}^2\right], \tag{6.12}$$

where we have used the fact that N is a zero-mean Gaussian variable independent of g^t, w^t. Taking expectation on both sides and using the convexity of $F(\cdot)$:

$$\mathbb{E}[F(w^t)] - F(w^*) \leqslant \frac{\mathbb{E}[\|w^t - w^*\|_{A^t}^2] - \mathbb{E}[\|w^{t+1} - w^*\|_{A^t}^2]}{2\alpha^t} + \frac{\alpha^t}{2}\mathbb{E}[\left\langle g^t, (A^t)^{-1} g^t \right\rangle] + \frac{\alpha^t}{2}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{6.13}$$

Applying telescope sum, we have

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)] - F(w^*)\right) \tag{6.14}$$

$$\leqslant \frac{\|w^1 - w^*\|_{A^1}^2}{2\alpha_1} + \sum_{t=2}^{T}\left(\frac{\mathbb{E}\left[\|w^t - w^*\|_{A^t}^2\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|w^t - w^*\|_{A^{t-1}}^2\right]}{2\alpha_{t-1}}\right) + \sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, (A^t)^{-1} g^t \right\rangle\right] + \sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{6.15}$$

Let $\alpha^t = \frac{\alpha}{\sqrt{t}}$,

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)] - F(w^*)\right) \tag{6.16}$$

$$\leqslant \frac{\|w^1 - w^*\|_{A^1}^2}{2\alpha} + \underbrace{\sum_{t=2}^{T}\frac{\mathbb{E}\left[\|w^t - w^*\|_{\sqrt{t}A^t - \sqrt{t-1}A^{t-1}}^2\right]}{2\alpha}}_{T_1} + \underbrace{\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\left\langle g^t, (A^t)^{-1} g^t \right\rangle\right]}_{T_2} + \sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{6.17}$$

Let $1 - \frac{\gamma}{t} \geqslant \beta^t \geqslant 1 - \frac{1}{t}$ for some $\gamma \in (0, 1]$ and $\sqrt{t}\epsilon^t \geqslant \sqrt{t-1}\epsilon^{t-1}$. We first bound T_1. Based on the relations between v^t and v^{t-1} and $\beta^t \geqslant 1 - \frac{1}{t}$, we can prove for any t, j

$$\sqrt{t}(A_j^t) = \sqrt{t}\left(\sqrt{v_j^t} + \epsilon^t\right) = \sqrt{t}\left(\sqrt{\beta^t v_j^{t-1} + (1-\beta^t)(\hat{g}_j^t)^2} + \epsilon^t\right) \geqslant \sqrt{(t-1)v_j^{t-1}} + \sqrt{t-1}\epsilon^{t-1}. \tag{6.18}$$

So for any j, t,

$$\sqrt{t}A_j^t \geqslant \sqrt{t-1}A_j^{t-1}. \tag{6.19}$$

Hence,

$$\mathbb{E}\left[\sum_{t=2}^{T} \|w^t - w^*\|^2_{\sqrt{t}A^t - \sqrt{t-1}A^{t-1}}\right] \tag{6.20}$$

$$= \mathbb{E}\left[\sum_{t=2}^{T}\sum_{j=1}^{d}(w_j^t - w_j^*)^2\left(\sqrt{tv_j^t} + \sqrt{t}\epsilon^t - \sqrt{(t-1)v_j^{t-1}} - \sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{6.21}$$

$$= \mathbb{E}\left[\sum_{j=1}^{d}\sum_{t=2}^{T}(w_j^t - w_j^*)^2\left(\sqrt{tv_j^t} + \sqrt{t}\epsilon^t - \sqrt{(t-1)v_j^{t-1}} - \sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{6.22}$$

$$\leqslant \mathbb{E}\left[\sum_{j=1}^{d} D^2 \sum_{t=2}^{T}\left(\sqrt{tv_j^t} + \sqrt{t}\epsilon^t - \sqrt{(t-1)v_j^{t-1}} - \sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{6.23}$$

$$= \mathbb{E}\left[\sum_{j=1}^{d} D^2\left(\sqrt{Tv_j^T} + \sqrt{T}\epsilon^T - \sqrt{v_j^1} - \epsilon^1\right)\right]. \tag{6.24}$$

We next bound T_2. We prove a variant of Lemma 4.1 in Mukkamala and Hein [218]. The major differences are in that we consider the stochastic case and estimating v^t on public data. We prove the following inequality by induction:

$$\sum_{t=1}^{T}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t} + \sqrt{t}\epsilon^t}\right] \leqslant \frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T} + \sqrt{T}\epsilon^T\right],\ j \in [d]. \tag{6.25}$$

For $t = 1$,

$$\mathbb{E}\left[\frac{(g_j^1)^2}{\sqrt{v_j^t} + \epsilon^1}\right] = \mathbb{E}\left[\frac{(g_j^1)^2}{\sqrt{(1-\beta^1)(\hat{g}_j^1)^2} + \epsilon^1}\right] \tag{6.26}$$

$$\leqslant \mathbb{E}\left[\frac{(\hat{g}_j^1)^2}{a\sqrt{(1-\beta^1)(\hat{g}_j^1)^2} + \epsilon^1}\right] \tag{6.27}$$

$$\leqslant \mathbb{E}\left[\frac{\sqrt{(1-\beta^1)(\hat{g}_j^1)^2}+\epsilon^1}{a(1-\beta^1)}\right]. \tag{6.28}$$

Suppose that the conclusion holds when $t = T-1$, i.e., for any $j \in [d]$,

$$\sum_{t=1}^{T-1}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t}+\sqrt{t}\epsilon^t}\right] \leqslant \frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{(T-1)v_j^{T-1}}+\sqrt{T-1}\epsilon^{T-1}\right]. \tag{6.29}$$

In addition, combined with the fact that $v_j^T = \beta^T v_j^{T-1} + (1-\beta^T)(\hat{g}_j^T)^2$ and $\sqrt{T}\epsilon^T \geqslant \sqrt{T-1}\epsilon^{T-1}$, we have

$$\begin{aligned}
\sqrt{(T-1)v_j^{T-1}}+\sqrt{T-1}\epsilon^{T-1} &\leqslant \sqrt{\frac{(T-1)v_j^T}{\beta^T}-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}}+\sqrt{T}\epsilon^T && (6.30)\\
&\leqslant \sqrt{Tv_j^T-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}}+\sqrt{T}\epsilon^T && (6.31)\\
&\leqslant \sqrt{Tv_j^T}-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}+\sqrt{T}\epsilon^T && (6.32)\\
&\leqslant \sqrt{Tv_j^T}+\sqrt{T}\epsilon^T-\frac{a(T-1)(1-\beta^T)(g_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}. && (6.33)
\end{aligned}$$

The third inequality comes from $\sqrt{a-b} \leqslant \sqrt{a}-\frac{b}{2\sqrt{a}}$ $(a \geqslant b)$ by letting a be Tv_j^T and b be $\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}$. Hence

$$\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t}+\sqrt{t}\epsilon^t}\right] && (6.34)\\
&\leqslant \frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T-\frac{a(T-1)(1-\beta^T)(g_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}\right]+\mathbb{E}\left[\frac{(g_j^T)^2}{\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T}\right] && (6.35)\\
&\leqslant \frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]+\left(1-\frac{(2-\gamma)(T-1)(1-\beta^T)}{\gamma\beta^T}\right)\mathbb{E}\left[\frac{(g_j^T)^2}{\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T}\right] && (6.36)\\
&\leqslant \frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]. && (6.37)
\end{aligned}$$

We then bound T_2 as follows.

$$\mathbb{E}\left[\sum_{t=1}^{T}\frac{\alpha^t}{2}\sum_{j=1}^{d}\frac{(g_j^t)^2}{\sqrt{v_j^t+\epsilon^t}}\right]=\frac{\alpha}{2}\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{d}\frac{(g_j^t)^2}{\sqrt{tv_j^t+\sqrt{t}\epsilon^t}}\right]\leqslant\frac{\alpha}{2}\sum_{j=1}^{d}\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]. \tag{6.38}$$

Noting that

$$\frac{\left\|w^1-w^*\right\|_{A^1}^2}{2\alpha}\leqslant\frac{D^2}{2\alpha}\sum_{j=1}^{d}\left(\sqrt{v_j^1}+\epsilon^1\right), \tag{6.39}$$

combined with the bounds of T_1, T_2 yields

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)-F(w^*)]\right)\leqslant\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right], \tag{6.40}$$

which implies that

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F(w^*) \tag{6.41}$$

$$\leqslant\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\frac{1}{T}\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]+\frac{1}{T}\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\|N\|_{A^t}^2\right] \tag{6.42}$$

$$\leqslant\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\frac{1}{\sqrt{T}}\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{v_j^T}+\epsilon^T\right]+\frac{\alpha}{\sqrt{T}}\max_{t\in[T]}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{6.43}$$

The first term is standard for the RMSprop optimizer, and the last term is due to noise added to ensure differential privacy. To guarantee overall (ε,δ)-differential privacy by running T total iterations, we set $\sigma^2=O\left(\frac{b^2T\log(1/\delta)}{n^2\varepsilon^2}\right)$ and $T=O\left(\frac{n^2\varepsilon^2}{\max_{t\in[T]}\mathbb{E}\left[\sum_{j=1}^{d}A_j^t\right]\log(1/\delta)}\right)$. The convergence rate becomes

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F(w^*)\leqslant O\left(\frac{\sqrt{\max_{t\in[T]}\mathbb{E}\left[\sum_{j=1}^{d}A_j^t\right]\log(1/\delta)}}{n\varepsilon}\right). \tag{6.44}$$

6.5.1.1 Proof for Theorem 17 (Fix A^t Before Training)

Denote the side information as a fixed A at any iteration t. Similar as previous analysis, setting a decaying learning rate $\alpha^t=\frac{\alpha}{\sqrt{t}}$, we have

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)]-F(w^*)\right) \tag{6.45}$$

$$\leqslant \frac{\|w^1 - w^*\|_A^2}{2\alpha} + \underbrace{\sum_{t=2}^{T} \frac{\mathbb{E}\left[\|w^t - w^*\|^2_{\sqrt{t}A - \sqrt{t-1}A}\right]}{2\alpha}}_{T_1} + \underbrace{\sum_{t=1}^{T} \frac{\alpha}{2\sqrt{t}} \mathbb{E}\left[\left\langle g^t, (A)^{-1} g^t \right\rangle\right]}_{T_2}$$

$$+ \sum_{t=1}^{T} \frac{\alpha}{2\sqrt{t}} \mathbb{E}\left[\|N\|_A^2\right]. \tag{6.46}$$

To bound T_1, we have

$$\sum_{t=2}^{T} \mathbb{E}\left[\|w^t - w^*\|^2_{\sqrt{t}A - \sqrt{t-1}A}\right] = \mathbb{E}\left[\sum_{t=2}^{T}\sum_{j=1}^{d} (w_j^t - w_j^*)^2 \left((\sqrt{t} - \sqrt{t-1})(A_j)\right)\right] \tag{6.47}$$

$$\leqslant O\left(\sqrt{T}\sum_{j=1}^{d} A_j\right) - \left\|w^1 - w^*\right\|_A^2. \tag{6.48}$$

We consider T_2 next. From the assumptions on the clipping bound,

$$R := \max_{j,t} \frac{\mathbb{E}\left[(g_j^t)^2\right]}{A_j^2} \leqslant C^2. \tag{6.49}$$

Then

$$\sum_{t=1}^{T}\sum_{j=1}^{d} \frac{\alpha}{2\sqrt{t}A_j} \mathbb{E}\left[(g_j^t)^2\right] \leqslant \sum_{t=1}^{T}\sum_{j=1}^{d} \frac{\alpha}{2\sqrt{t}} R A_j \leqslant \sqrt{T}\alpha R \sum_{j=1}^{d} A_j. \tag{6.50}$$

Therefore, we obtain

$$\min_{t\in[T]} \mathbb{E}[F(w^t)] - F(w^*) \leqslant O\left(\frac{1}{\sqrt{T}}\sum_{j=1}^{d} A_j + \frac{\alpha R}{\sqrt{T}}\sum_{j=1}^{d} A_j + \frac{\alpha}{\sqrt{T}} \mathbb{E}\left[\|N\|_A^2\right]\right). \tag{6.51}$$

6.5.2 Proof for Theorem 18 (Non-convex and smooth cases)

We use the same ϵ at each iteration. Let $\nabla_j F(w)$ denote the j-th coordinate of $\nabla F(w)$ for any w. Based on the L-smoothness of $F(\cdot)$,

$$F(w^{t+1}) \leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \left(\nabla_j F(w^t) \cdot \left(\frac{g_j^t}{\sqrt{v_j^t} + \epsilon} + N\right)\right)$$

$$+ \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \left(\frac{(g_j^t)^2}{\left(\sqrt{v_j^t} + \epsilon\right)^2} + N^2 + \frac{g_j^t}{\sqrt{v_j^t} + \epsilon} \cdot 2N\right), \tag{6.52}$$

where $N \sim \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2)$ and $\mathbb{E}[N^2] = \frac{\sigma^2 C^2}{b^2}$. Taking expectation conditioned on w^t on both sides gives

$$\mathbb{E}_t[F(w^{t+1})] \leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \nabla_j F(w^t) \mathbb{E}_t \left[\frac{g_j^t}{\sqrt{v_j^t} + \epsilon} \right] + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \mathbb{E}_t \left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t} + \epsilon\right)^2} \right] + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2. \tag{6.53}$$

The following proof is extended from that of Theorem 1 in Zaheer et al. [318].

$$\mathbb{E}_t[F(w^{t+1})] \leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \nabla_j F(w^t) \mathbb{E}_t \left[\frac{g_j^t}{\sqrt{v_j^t} + \epsilon} - \frac{g_j^t}{\sqrt{\beta v_j^{t-1}} + \epsilon} + \frac{g_j^t}{\sqrt{\beta v_j^{t-1}} + \epsilon} \right] + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \mathbb{E}_t \left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t} + \epsilon\right)^2} \right] + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2 \tag{6.54}$$

$$\leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}} + \epsilon} + \alpha^t \sum_{j=1}^{d} \left| \nabla_j F(w^t) \right| \left| \mathbb{E}_t \left[\frac{g_j^t}{\sqrt{v_j^t} + \epsilon} - \frac{g_j^t}{\sqrt{\beta v_j^{t-1}} + \epsilon} \right] \right| + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \mathbb{E}_t \left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t} + \epsilon\right)^2} \right] + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2. \tag{6.55}$$

Further,

$$\frac{g_j^t}{\sqrt{v_j^t} + \epsilon} - \frac{g_j^t}{\sqrt{\beta v_j^{t-1}} + \epsilon} \leqslant \left| g_j^t \right| \left| \frac{1}{\sqrt{v_j^t} + \epsilon} - \frac{1}{\sqrt{\beta v_j^{t-1}} + \epsilon} \right| \tag{6.56}$$

$$= \frac{\left| g_j^t \right|}{\left(\sqrt{v_j^t} + \epsilon\right)\left(\sqrt{\beta v_j^{t-1}} + \epsilon\right)} \frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{v_j^t} + \sqrt{\beta v_j^{t-1}}\right)} \tag{6.57}$$

$$= \frac{\left| g_j^t \right|}{\left(\sqrt{v_j^t} + \epsilon\right)\left(\sqrt{\beta v_j^{t-1}} + \epsilon\right)} \frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{\beta v_j^{t-1} + (1-\beta)(\hat{g}_j^t)^2} + \sqrt{\beta v_j^{t-1}}\right)} \tag{6.58}$$

$$\leqslant \frac{1}{\left(\sqrt{v_j^t} + \epsilon\right)\left(\sqrt{\beta v_j^{t-1}} + \epsilon\right)} \frac{\sqrt{1-\beta}}{\sqrt{a}} (g_j^t)^2 \tag{6.59}$$

$$\leqslant \frac{1}{\left(\sqrt{\beta v_j^{t-1}} + \epsilon\right)\epsilon} \frac{\sqrt{1-\beta}}{\sqrt{a}} (g_j^t)^2. \tag{6.60}$$

We have used the observation $\frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{\beta v_j^{t-1}} + (1-\beta)(\hat{g}_j^t)^2 + \sqrt{\beta v_j^{t-1}}\right)} \leqslant \sqrt{1-\beta}\hat{g}_j^t$, and $\hat{g}_j^t \leqslant \frac{1}{\sqrt{a}} g_j^t$.

From L-smoothness of $F(\cdot)$ which implies that $\|\nabla F(u) - \nabla F(v)\| \leqslant L\|u - v\|$ for any $u, v \in \mathbb{R}^d$, and Assumption 2, it is easy to see there exists a constant B such that $|\nabla_j F(w)| \leqslant B$ for any $j \in [d]$.

$$\begin{aligned}
\mathbb{E}_t[F(w^{t+1})] &\leqslant F(w^t) - \alpha^t \sum_{j=1}^d \frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}} + \epsilon} + \frac{\alpha^t B\sqrt{1-\beta}}{\epsilon\sqrt{a}} \sum_{j=1}^d \mathbb{E}_t \left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}} + \epsilon}\right] \\
&\quad + \frac{(\alpha^t)^2 Ld}{2} \sum_{j=1}^d \mathbb{E}_t \left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t} + \epsilon\right)^2}\right] + \frac{(\alpha^t)^2 Ld}{2b^2}\sigma^2 C^2 \qquad (6.61) \\
&\leqslant F(w^t) - \alpha^t \sum_{j=1}^d \frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}} + \epsilon} + \frac{\alpha^t B\sqrt{1-\beta}}{\epsilon\sqrt{a}} \sum_{j=1}^d \mathbb{E}_t \left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}} + \epsilon}\right] \\
&\quad + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^d \mathbb{E}_t \left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}} + \epsilon}\right] + \frac{(\alpha^t)^2 Ld}{2b^2}\sigma^2 C^2, \qquad (6.62)
\end{aligned}$$

where the last inequality holds due to $\left(\sqrt{v_j^t} + \epsilon\right)^2 \geqslant \epsilon\left(\epsilon + \sqrt{v_j^t}\right) \geqslant \epsilon\left(\epsilon + \sqrt{\beta v_j^{t-1}}\right)$. Lemma 1 in Zaheer et al. [318] proves that $\mathbb{E}_t\left[(g_j^t)^2\right] \leqslant \frac{\tau_j^2}{b} + (\nabla_j F(w^t))^2$ where τ_j^2 is the variance bound of the j-th coordinate, i.e., $\mathbb{E}\left[\|g_j^t - \nabla_j F(w^t)\|^2\right] \leqslant \tau_j^2$. Plugging this inequality into Eq. (6.62), combined with $\frac{L\alpha^t}{2\epsilon} \leqslant \frac{1}{4}$ and $\frac{B\sqrt{1-\beta}}{\sqrt{a}\epsilon} \leqslant \frac{1}{4}$, we obtain

$$\begin{aligned}
\mathbb{E}_t[F(w^{t+1})] \leqslant F(w^t) - \frac{\alpha^t}{2(\sqrt{\beta}B + \epsilon)}\|\nabla F(w^t)\|^2 + \left(\frac{\alpha^t B\sqrt{1-\beta}}{\sqrt{a}\epsilon^2} + \frac{L(\alpha^t)^2}{2\epsilon^2\sqrt{\beta}}\right)\frac{\sum_{j\in[d]}\tau_j^2}{b} \\
+ \frac{(\alpha^t)^2 Ld}{2b^2}\sigma^2 C^2.
\end{aligned}$$

Taking expectation on both sides and applying the telescope sum yields

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}[\|\nabla F(w^t)\|^2] \leqslant O\left(\frac{1}{T}\right) + O\left(\frac{\|\tau^2\|_1}{b}\right) + O\left(\frac{\sigma^2}{b^2}\right) \tag{6.63}$$

6.5.2.1 Proof for Theorem 19 (Fix A before training)

Due to L-smoothness of $F(\cdot)$, we have

$$F(w^{t+1}) \leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \left(\nabla_j F(w^t) \cdot \left(\frac{g_j^t}{A_j} + N \right) \right) + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \left(\frac{(g_j^t)^2}{A_j^2} + N^2 + \frac{g_j^t}{A_j} \cdot 2N \right),$$

where $N \sim \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2)$. Taking expectation conditioned on w^t on both sides gives

$$\mathbb{E}_t \left[F(w^{t+1}) \right] \tag{6.64}$$

$$\leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \frac{1}{A_j^2} \mathbb{E}_t \left[(g_j^t)^2 \right] + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2 \tag{6.65}$$

$$\leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \frac{1}{A_j^2} \left(\frac{\tau_j^2}{b} + (\nabla_j F(w^t))^2 \right) + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2 \tag{6.66}$$

$$\leqslant F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{1}{A_j} \left(\frac{\tau_j^2}{b} + (\nabla_j F(w^t))^2 \right) + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2 \tag{6.67}$$

$$\leqslant F(w^t) - \frac{\alpha^t}{2} \left\| \nabla F(w^t) \right\|_{A^{-1}}^2 + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{\tau_j^2}{A_j b} + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2. \tag{6.68}$$

The last inequality is due to $\alpha^t \leqslant \frac{\epsilon}{L}$. Taking expectation on both sides yields

$$\mathbb{E}\left[F(w^{t+1}) \right] \leqslant \mathbb{E}\left[F(w^t) \right] - \frac{\alpha^t}{2} \mathbb{E}[\|\nabla F(w^t)\|_{A^{-1}}^2] + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{\tau_j^2}{A_j b} + \frac{(\alpha^t)^2 L d}{2b^2} \sigma^2 C^2. \tag{6.69}$$

Similarly, by rearranging terms and applying telescope sum, we obtain

$$\frac{1}{T} \sum_{t \in [T]} \mathbb{E}\left[\|\nabla F(w^t)\|_{A^{-1}}^2 \right] \leqslant O\left(\frac{1}{T} \right) + O\left(\frac{\tau^2}{A} \cdot \frac{1}{b} \right) + O\left(\frac{\sigma^2}{b^2} \right). \tag{6.70}$$

6.6 Experimental Details

Pseudo Code of Some Algorithms. For completeness, we present the full baseline DP-Adam algorithm in Algorithm 9 and `AdaDPS` extended to federated learning in Algorithm 10.

Datasets and Models. We consider a diverse set of datasets and tasks.

Algorithm 9 DP-Adam [332]

Input: T, batch size b, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, $v^0 = \mathbf{0}$, $m^0 = \mathbf{0}$, small constant vector $\epsilon^t \in \mathbb{R}^d$, learning rate α^t, moving average parameters β_1, β_2

for $t = 1, \cdots, T-1$ **do**

Uniformly randomly sample a mini-batch B with size b from private training data

Get individual gradients for sample $i \in B$:

$$g^{i,t} \leftarrow \nabla f(x^i; w^t)$$

Private gradients using Gaussian mechanism:

$$\tilde{g}^t \leftarrow \frac{1}{b} \sum_{i \in B} \text{clip}\left(g^{i,t}, C\right) + \frac{1}{b} \mathcal{N}(0, \sigma^2 C^2)$$

Update first and second moment estimates as

$$m^t \leftarrow \beta_1 m^{t-1} + (1-\beta_1)\tilde{g}^t$$
$$v^t \leftarrow \beta_2 v^{t-1} + (1-\beta_2)(\tilde{g}^t)^2$$

Update the model parameter w as

$$w^{t+1} \leftarrow w^t - \alpha^t \frac{m^t/(1-\beta_1^t)}{\sqrt{v^t/(1-\beta_2^t)} + \epsilon^t},$$

where β_1^t, β_2^t denotes β_1, β_2 to the power of t (with slight abuse of notations)

end for

return w^T

- **StackOverflow** [21] consists of posts on the Stack Overflow website, where the task is tag prediction (500-class classification). We randomly subsample 246,092 samples from the entire set. There are 10,000 input features in StackOverflow, resulting in more than 5 million learnable parameters in a logistic regression model.
- **IMDB** [199] is widely used for for binary sentiment classification of movie reviews, consisting of 25,000 training and 25,000 testing samples. We study two models on IMDB: logistic regression and neural networks (with LSTM) with 20,002 and 706,690 parameters, respectively. For logistic regression, we set the vocabulary size to 10,000 and consider two sets of commonly-used features separately: bag-of-words (BoW) and TF-IDF values.
- **MNIST** [165] images with a deep autoencoder model (for image reconstruction) which

Algorithm 10 `AdaDPS` applied to federated learning

Input: T communication rounds, b selected devices each round, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, non-sensitive side information A, number of local iterations s, local learning rate η^t

for $t = 1, \cdots, T-1$ **do**

Server uniformly selects a subset B of b devices and sends the current global model w^t to them Each device $i \in B$ sets the local model to be the current global model:

$$w^{i,0} \leftarrow w^t$$

Each device $i \in B$ runs adaptive mini-batch SGD locally with side information A to obtain model updates:

for $j = 0, \cdots, s$ **do**

$$w^{i,j+1} \leftarrow w^{i,j} - \eta^t \frac{\nabla f(w^{i,j})}{A}$$

end for

And then privatize model updates:

$$\Delta^{i,t} \leftarrow w^{i,s+1} - w^{i,0}$$
$$\tilde{\Delta}^{i,t} \leftarrow \text{clip}(\Delta^{i,t}, C) + \mathcal{N}(0, \sigma^2 C^2)$$

Each device $i \in B$ sends $\tilde{\Delta}^{i,t}$ to the server

Server updates the global model as:

$$w^{t+1} \leftarrow w^t + \frac{1}{b} \sum_{i \in B} \tilde{\Delta}^{i,t}$$

end for

return w^T

has the same architecture as that in previous works [318] (containing more than 2 million parameters). The loss is reconstruction error measured as the mean squared distance in the pixel space. We scale each input feature to the range of $[0, 1]$.

Hyperparameter Tuning. We detail our hyperparameter tuning protocol and the hyperparameter values here. Our code is publicly available at `github.com/litian96/AdaDPS`.

- For non-private training experiments, we fix the mini-batch size to 64, and tune fixed learning rates by performing a search over $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2\}$ separately for all methods on validation data. We do not use momentum for AdaS (i.e., applying the idea of preconditioning of `AdaDPS` without privatization) for all

centralized training experiments.

- For differentially private training, the δ values in the privacy budget are always inverse of the number of training samples. We fix the noise multiplier σ for each dataset, tune the clipping threshold, and compute the final ε values. Specifically, the σ values are 1, 1, and 0.95 for IMDB (convex), IMDB (LSTM), and StackOverflow; 1 and 0.75 for MNIST (autoencoder). The clipping threshold C (in Algorithm 8) is tuned from $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 3\}$, jointly with tuning the (fixed) learning rates. The number of micro-batches is 16 for all related experiments, and the mini-batch size is 64 (i.e., we privatize each gradient averaged over 4 individual ones to speed up computation).
- For federated learning experiments, we fix server-side learning rate to be 1 (i.e., simply applying the unscaled average of noisy model updates in Line 9 of Algorithm 10), and apply server-side momentum [235] with a moving average parameter 0.9 for all methods in the left sub-figure in Figure 6.7. The number of local epochs is set to 1 for all runs, and the local mini-batch size is 100.

The tuned hyperparameter values (clipping threshold C, learning rate) for private training are summarized in Table 6.5 below.

Datasets	**DP-SGD**	**DP-Adam**	**DP-RMSProp**	`AdaDPS` **(RMSProp)**	`AdaDPS` **(Adam)**
IMDB (convex)	(0.1, 1)	(0.02, 0.1)	(0.05, 0.1)	(2, 0.5)	(2, 1)
IMDB (LSTM)	(0.1, 0.1)	(0.1, 0.001)	(0.1, 0.001)	(0.2, 0.1)	(0.2, 0.05)
StackOverflow (linear)	(0.1, 1)	(0.1, 0.01)	(0.2, 0.01)	(1, 0.5)	(1, 0.5)
MNIST (autoencoder)	(0.05, 0.5)	(0.01, 0.001)	(0.01, 0.001)	(3, 0.005)	(3, 0.005)
MNIST (classification)	(0.5, 0.01)	(0.5, 0.001)	(0.5, 0.001)	(2, 0.005)	(2, 0.005)

Table 6.5: Major hyperparameter values (learning rate and clipping threshold C) used in private experiments for all datasets. 'IMDB (convex)' is IMDB (BoW features) on a logistic regression model. StackOverflow results are for centralized training. The noise multiplier σ values in these four tasks are 1, 1, 0.95, 1, respectively, resulting in ε values being 1.5, 2.8, 0.84, and 1.6.

6.7 Additional Results

6.7.1 Additional Baselines

Other Private Adaptive Optimization Baselines. In the main text, we mainly compare `AdaDPS` with DP-Adam (summarized in Algorithm 9). There are other possible baselines similar as DP-Adam, by replacing Adam with other adaptive methods, resulting in DP-AdaGrad and DP-RMSProp. This line of differentially private optimizers has similar empirical performance as DP-Adam, as shown in the results in Table 6.6 below (using DP-RMSProp as an example).

Datasets	Metrics	DP-SGD	DP-Adam	DP-RMSProp	AdaDPS (RMSProp)	AdaDPS (Adam)
IMDB (convex)	accuracy	0.63	0.69	0.67	0.80	0.80
IMDB (LSTM)	accuracy	0.70	0.69	0.69	0.73	0.73
StackOverflow (linear)	accuracy	0.28	0.30	0.31	0.40	0.40
MNIST (autoencoder)	loss ($\times 100$)	5.013	3.697	3.636	3.566	3.443
MNIST (classification)	accuracy	0.9273	0.9333	0.9314	0.9377	0.9541

Table 6.6: Full comparisons between AdaDPS and private adaptive optimization methods. The evaluation metrics are reported on test data. 'IMDB (convex)' is IMDB (BoW features) on a logistic regression model. For (ε, δ)-differential privacy, the ε values of experiments in the five rows are 1.5, 2.8, 0.84, 1.6, and 1.25, respectively, and the δ values are the inverse of the number of training samples (as mentioned in the main text).

Using Public Data for Pretraining. Another possible way of leveraging public data to improve privacy/utility tradeoffs is to pretrain on them. However, this would give only limited performance improvement if the amount of public data is very small. In the main text, when needed, AdaDPS randomly samples 1% training data as public data. Under this setup, we empirically compare AdaDPS with the pre-training baseline (denoted as DP-SGD w/ warm start). From Table 6.7, we see that AdaDPS outperforms it by a large margin.

Datasets	Metrics	DP-SGD	DP-SGD w/ warm start	AdaDPS w/ public
IMDB (convex)	accuracy	0.63	0.73	0.80
StackOverflow	accuracy	0.28	0.33	0.40
MNIST (autoencoder)	loss	0.050	0.049	0.036

Table 6.7: Compare AdaDPS with an additional baseline of DP-SGD pre-trained on public data on three datasets. For 'DP-SGD w/ warm start', we first train on public data for 10 epochs via adaptive methods (RMSProp), and then run DP-SGD on private data starting from that initialization.

DP-Adam with Public Data. In the main text (Section 6.4.1.1), we discuss the DP-R-Pub. baseline based on the RMSProp method with the preconditioner estimated on public data. Similarly, one can also apply such clean preconditioners in DP-Adam updates, resulting in another baseline which we call DP-Adam-Pub. The main differences between DP-Adam-Pub and DP-R-Pub are that DP-Adam-Pub additionally considers momentum. Formally, the updates of w^t is as follows:

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i\in B} \text{clip}\left(g^{i,t}, C\right) + \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2), \quad \hat{g}^t \leftarrow \mathbb{E}_x\left[\nabla f(x; w^t)\right] \text{ for } x \in x_{\text{pub}},$$
$$m^t \leftarrow \beta_2 m^t + (1-\beta_2)\left(\beta_1 \tilde{g}^t + (1-\beta_1)\hat{g}^t\right), \quad v^t \leftarrow \beta_3 v^t + (1-\beta_3)(\hat{g}^t)^2,$$

$$w^{t+1} \leftarrow w^t - \alpha^t \frac{m^t/(1-(\beta_2)^t)}{\sqrt{v^t/(1-(\beta_3)^t)} + \epsilon},$$

where $\beta_1, \beta_2, \beta_3, \epsilon$ are small constants.

Datasets	**Metrics**	**DP-SGD**	**DP-Adam-Pub**	AdaDPS w/ public
IMDB (convex)	accuracy	0.63	0.74	0.80
StackOverflow	accuracy	0.28	0.31	0.40
MNIST (autoencoder)	loss	0.050	0.064	0.036

Table 6.8: Results of comparing AdaDPS with to DP-Adam-Pub (i.e., DP-Adam using clean preconditioners estimated on public data).

6.7.2 Side Information in Non-Private Training

In the main text, we mainly focus on private optimization. It is expected that side information (even without the assist of public data) would also be beneficial in non-private settings, which could serve as a simple alternative to adaptive methods. We report results in Table 6.9.

Datasets	**Metrics**	**SGD**	**Adam**	AdaS (w/ public)	AdaS (w/o public)
IMDB (convex)	accuracy	0.66	0.88	0.88	0.88
IMDB (LSTM)	accuracy	0.88	0.88	0.88	0.88
StackOverflow (linear)	accuracy	0.38	0.64	0.64	0.64
MNIST (autoencoder)	loss ($\times 100$)	5.013	1.151	1.805	—

Table 6.9: Performance of each method in *non-private* training. We see that AdaS can match the performance of Adam in non-private settings.

6.7.3 Effects of Public Data Size

We further study the effects of public data size. Only a very small set of public data (even 0.04% the size of private training data) can provide good preconditioner estimates.

Datasets	*upper bound*	AdaDPS 1% public	AdaDPS 5$\times$ less	AdaDPS 25$\times$ less
IMDB (convex)	0.82	0.80	0.80	0.75
StackOverflow	0.41	0.40	0.40	0.39

Table 6.10: Effects of public data sizes.

6.8 DP^2: Private Adaptive Optimization Without Side Information

We now introduce our DP^2 framework. While we discuss DP^2 in the context of a particular adaptive method (RMSProp), we note that the approach is method-agnostic in that it can generally be applied to any private adaptive optimization method where preconditioners are calculated at each iteration. As an initial step towards understanding the algorithm, we first investigate the effects of delayed preconditioners in non-private training in Section 6.8.1. We then explain how to apply this idea to construct less noisy preconditioners from prior gradients in private training in Section 6.8.2.

6.8.1 Delayed Preconditioners in Non-Private Settings

Adaptive methods use preconditioners to adapt to gradient geometry, effectively resulting in coordinate-wise learning rates. This can be advantageous for many applications, especially those with sparse gradients or non-uniform stochastic noise [e.g., 117, 207, 235, 320]. One of the key design choices of DP^2 is to update preconditioners less frequently and use the average of past gradients to reduce noise. Our observation is that a wide range of learning problems are tolerant to the staleness of preconditioners. In this subsection, we validate this empirically on the benchmark datasets considered throughout this work.

There are potentially many ways that one could instantiate the idea of delayed preconditioner computation in adaptive optimization. Here we consider a specific algorithm, which is the exact non-private version of our proposed DP^2 framework (Algorithm 11) introduced in later sections. The basic idea is to alternate between s steps of SGD and s steps of an adaptive method (for simplicity we assume RMSProp as the adaptive algorithm), where s is a constant larger than 1. Each time we switch from SGD to RMSProp, we average s past SGD gradients and use the average to update the preconditioner. The preconditioner will be used in subsequent RMSProp updates (thus being stale). As motivation for DP^2, we empirically show that RMSProp with delayed preconditioners achieves almost the same optimization performance as RMSProp (Figure 6.8).

We note that the idea of delayed preconditioning has been briefly discussed in prior work [104] for the purpose of speeding up the computation of adaptive optimization in non-private training. Unlike this prior work, we focus on the goal of reducing noise in private training, propose an alternative method for using stale preconditioners that is more amenable to differential privacy, and analyze our method in both convex and non-convex settings.

6.8.2 Constructing Delayed Preconditioners with Reduced Noise

Without access to public data or other side information, prior works typically update preconditioners based on noisy gradients at each iteration [332]. For instance, a natural way to privatize RMSProp is to update the preconditioner $v \in \mathbb{R}^d$ as $v \leftarrow \beta v + (1-\beta)(\tilde{g})^2$

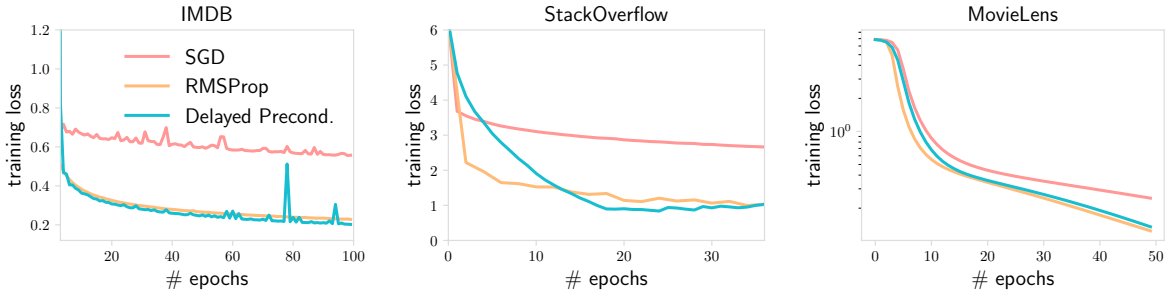


Figure 6.8: In non-private training, RMSProp with delayed preconditioners achieves similar training loss as standard RMSProp across all datasets. Final test accuracies are presented in Section 6.10.1. This observation provides motivation for our proposed DP^2 framework for private training (Section 6.8.2).

where $\beta \in (0, 1)$ is a moving average constant, and $\tilde{g} \in \mathbb{R}^d$ is the noisy gradient output by some standard privacy mechanism (e.g., the Gaussian mechanism).[3] However, a drawback to this is that the noise gets accumulated at each iteration, making adaptive methods significantly less effective [187].

Inspired by the observation that problems can be tolerant to the staleness of preconditioners (Figure 6.8), we propose to update the preconditioners less frequently to reduce noise. For instance, we update v every s steps using some aggregate function of s recent private gradients from DP-SGD. During iterations where v is not updated, we simply apply the most recent (stale) v to precondition the gradients. In order to mitigate the noise, we *average* over these s gradients to form a pseudo-gradient g, which can be plugged into arbitrary adaptive optimization algorithms. Note that the privacy noise variance will be reduced s times if we average s Gaussian random variables (i.e., the DP noise).

DP^2 is summarized in Algorithm 11. For simplicity of presentation, we assume RMSProp as the adaptive method (denoted as DP^2-RMSProp) throughout this section. However, our framework can be generally applied to other common adaptive methods (see Appendices 6.13.3 and 6.14). The high-level idea is to alternate between s_1 steps of private SGD and s_2 private RMSProp steps, and use averages of s_1 SGD gradients (i.e., average of the accumulator $G \in \mathbb{R}^d$) to update the preconditioner v. Next, we discuss some key components of our algorithm.

Order of Privatization and Preconditioning. Given a private preconditioner v, there are generally two choices to perform adaptive optimization over the raw gradients $\{g^{i,t}\}_{i \in B}$ generated from mini-batch B at the t-th iteration.

1. First privatize gradients with clipping threshold C_1, then precondition noisy gradients

[3] We consider the practical diagonal (as opposed to matrix) form of adaptive methods.

with $\sqrt{v} + \epsilon$ where ϵ is a small constant:

$$\tilde{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(g^{i,t}, C_1\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C_1^2\right)\right) / \left(\sqrt{v} + \epsilon\right)$$

2. First precondition gradients with $\sqrt{v} + \epsilon$, then privatize the output with clipping threshold C_2:

$$\tilde{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(g^{i,t} / \left(\sqrt{v} + \epsilon\right), C_2\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C_2^2\right)\right)$$

The difference is that the privacy noise in the first choice may be scaled in an undesired direction, as $\frac{\mathcal{N}(\mathbf{0}, \sigma^2 C^2)}{\sqrt{v}+\epsilon}$ with a *less noisy* estimated $\sqrt{v}$ (perfect estimation removing all privacy noise in the extreme case) would amplify the noise $\mathcal{N}(\mathbf{0}, \sigma^2 C^2)$ on informative coordinates (i.e., coordinates with smaller preconditioner values), which is consistent with the argument made in Li et al. [187]. We empirically compare the two options and show that the latter gives better performance (Section 6.10.3).

It is critical to *average* noisy gradients to construct a cleaner estimate of the preconditioner (Line 5 and 10 in Algorithm 11) and apply it for adaptive optimization (Line 9). As these two steps access raw gradients twice, we need to privatize them separately. Unfortunately, the privacy budget would accumulate with each query to the raw training data. Hence, we use the private SGD gradients for both the model update and the preconditioner estimation. This results in a hybrid method that alternates between private SGD and private adaptive optimization steps. Note that to get an unbiased estimate of the true delayed preconditioners, we can correct the bias in $(G^t/s_1)^2$ (Line 5) by subtracting the privacy noise variance term $\frac{\sigma^2 C^2}{s_1 b^2}$ out of $(G^t/s_1)^2$. But this value is usually very small and negligible in practice. While in principle, non-adaptive and adaptive updates can take different numbers of consecutive iterations, in our empirical evaluation, we simply set $s_1 = s_2$, and find that this works reasonably well across all datasets (Section 6.10).

Privacy Guarantees. From Algorithm 11, we see that at each iteration, we access raw data and pass them through the privacy barrier *once* (Line 9) to generate private gradients $\tilde{g}^t$ with the same noise multiplier σ and batch size b, and the preconditioner only accumulates already differentially private gradients. Since the final model is a composition of these private releases (noisy gradients), Algorithm 11 (or $\texttt{DP}^2$ in general) achieves the same privacy guarantees as standard DP-SGD training under the same training settings. For completeness, we formally state the privacy guarantee below.

Theorem 20 (Privacy guarantee of Algorithm 11 [4])**.** *There exist constants c_1 and c_2 such that for any $\varepsilon < c_1 b^2 T/n^2$, Algorithm 11 is (ε, δ)-differentially private for any $\delta > 0$ if $\sigma \geqslant c_2 \frac{b\sqrt{T \log(1/\delta)}}{n\varepsilon}$.*

In practice, we use Rényi differential privacy (RDP) for the subsampled Gaussian mechanism accountant [213] to compute the actual ε's reported in the experiments (Section 6.10).

6.9 Convergence Analysis of DP^2

In this section, we analyze Algorithm 11 for both convex and non-convex problems. We aim to study the convergence properties of DP^2 and investigate the tradeoffs between delay and privacy noise. In doing so, key challenges are introduced by alternating between adaptive and non-adaptive updating and through the staleness of preconditioners.

6.9.1 Convex Cases

For convex functions, we define the optimal model w^* as $w^* \in \arg\min_w F(w)$. We reuse Assumption 2 and 3 in the previous section. Under the bounded stochastic gradient norm assumption, suppose the clipping does not happen, we have $\tilde{g}^t \leftarrow g^t + \mathcal{N}(0, \sigma^2 C^2/b^2)$, where $g^t := \frac{1}{b}\sum_{i\in B} g^{i,t}$. Without loss of generality, let $s_1{=}s_2$ in Algorithm 11. Our main convergence result is as follows (assuming t starts from 1).

Theorem 21 (Convergence of Algorithm 11 for convex problems)**.** *Let Assumptions 2 and 3 hold. Assume F is a convex function. Let the learning rate α^t be set as $\alpha^t \leftarrow \frac{\alpha\lfloor\frac{t}{2s}\rfloor+\lfloor\frac{t+s}{2s}\rfloor+1}{\sqrt{t}}$. After running Algorithm 11 for T iterations with $s = vT$ for a small constant $v \in (0, 1]$, we obtain*

$$\min_{t\in[T]} \mathbb{E}\big[F(w^t)\big] - F(w^*) \leqslant \frac{R^2+\kappa}{\alpha^{\lfloor\frac{1}{2v}\rfloor+\lfloor\frac{1+v}{2v}\rfloor}} \frac{1}{\sqrt{T}} \sum_{t\in T_v} \mathbb{E}\big[\|D^t\|_1\big] + \frac{1}{T}\sum_{t=1}^{T} \frac{\alpha^{\lfloor\frac{t}{2vT}\rfloor+\lfloor\frac{t+vT}{2vT}\rfloor}}{\sqrt{t}} \mathbb{E}[\|N^t\|^2_{D^t}],$$

where T_v denotes the iteration indices where we switch from private RMSProp steps to private SGD steps plus the last iteration, with cardinality $|T_v| = \lceil\frac{1}{2v}\rceil$, $N^t \sim \mathcal{N}(\mathbf{0}, \sigma^2C^2/b^2)$, and

$$\kappa \geqslant \max\left\{\alpha^2C^2, \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}\right\},\ \alpha = \min\left\{\epsilon, \frac{1}{\sqrt{M}+\epsilon}, 1\right\}\ \ \textit{where } M := C^2 + \frac{\sigma^2C^2}{sb^2}.$$

We defer all proofs to Appendix 6.11 and state simplified convergence results in Corollary 5. As we can see, the above upper bound relies on a critical metric $h(s)$ which is related to temporal gradient similarity and the amount of staleness s, formally defined as:

$$h(s) \geqslant \max_{t\in[T]} \frac{\mathbb{E}\left[\|g^t\|_1\right]}{\mathbb{E}\left[\left\|\frac{1}{s}G^{\lfloor\frac{t}{s}\rfloor s}\right\|_1\right] + d\epsilon} = \max_{t\in[T]} \frac{\mathbb{E}\left[\|g^t\|_1\right]}{\mathbb{E}\left[\frac{1}{s}\left\|\sum_{i=\lfloor\frac{t}{s}\rfloor s-s}^{\lfloor\frac{t}{s}\rfloor s-1} \tilde{g}^i\right\|_1\right] + d\epsilon},$$

where the expectation is taken with respect to all randomness in the algorithm, and $G^{\lfloor \frac{t}{s} \rfloor s} \in \mathbb{R}^d$ refers to the latest accumulator that is used to update v (Line 5 in Algorithm 11). A smaller $h(s)$ indicates better convergence. We see that the denominator of $h(s)$ can be decomposed into the average of past raw gradients and the average of random Gaussian noise. Intuitively, $h(s)$ tends to be smaller as gradients across the s iterations in $G^{\lfloor \frac{t}{s} \rfloor s}$ are more similar with the current gradient g^t in terms of the gradient norms. In Appendix 6.11.2, we show that an upper bound of $h(s)$ can be expressed as $c_1 + c_2 s$ where c_1, c_2 are two constants. We also visualize the value of $h(s)$ on the IMDB dataset in Figure 6.9, and show that (1) the values of $h(s)$ are consistently small across all delays, and (2) $h(s)$ increases as the s gets larger, which is consistent with the expression of s.

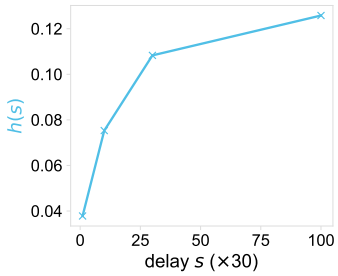


Figure 6.9: Visualization of $h(s)$ versus s on IMDB.

tradeoffs between Delay and Noise. Here we discuss how s affects convergence based on our analysis. Intuitively, larger s (larger delay) results in staler preconditioners, but introduces less noise due to private gradient averaging. In our convergence bound, there are several terms that depend on s (or v). Although this makes it difficult to derive a closed-form characterization of an optimal s, we can analyze the effects of s in simplified settings. In particular, examine the first term of the RHS of the convergence bound, let $\alpha = \frac{1}{\sqrt{M+\epsilon}} = \frac{1}{\sqrt{c_3 + \frac{c_4}{v} + \epsilon}}$ (where c_3, c_4 are two constants), and assume $\left\lfloor \frac{1}{2v} \right\rfloor + \left\lfloor \frac{1+v}{2v} \right\rfloor = \frac{1}{2v} + \frac{1+v}{2v} = \frac{2+v}{2v}$. Combined with $h(s)$, the dependence on v in $\frac{R^2+\kappa}{\alpha^{\left\lfloor \frac{1}{2v} \right\rfloor + \left\lfloor \frac{1+v}{2v} \right\rfloor}}$ can be expressed as $(c_1 + c_2 v)\left(\sqrt{c_3 + \frac{c_4}{v}} + \epsilon\right)^{\frac{2+v}{2v}}$. This suggests that there exists an optimal v that achieves the minimal value. In Section 6.10.1, we empirically study the effects of s across real-world datasets, and demonstrate that there exist specific ranges of s that provide favorable tradeoffs between delay and noise (Figure 6.12).

Corollary 5. *Let Assumptions 2 and 3 hold. Assume F is a convex function. Ignoring the constants, the convergence rate under learning rate $\alpha^t = O\left(\frac{1}{\sqrt{t}}\right)$ simplifies to*

$$\min_{t \in [T]} \mathbb{E}[F(w^t)] - F(w^*) \leqslant O\left(\frac{1}{\sqrt{T}} \max_{t \in T_s} \mathbb{E}\left[\|D^t\|_1\right]\right) + O\left(\frac{1}{T} \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \mathbb{E}\left[\|N^t\|^2_{D^t}\right]\right),$$

where T_s denotes the iteration indices where we switch from private RMSProp steps to private SGD steps plus the last iteration (thus having a constant cardinality) and $N^t \sim \mathcal{N}(\mathbf{0}, \sigma^2 C^2 / b^2)$.

At a high level, the first term is due to adaptive optimization using RMSProp, and the second term corresponds to the added privacy noise. Our $O\left(\frac{1}{\sqrt{T}}\right)$ rate is the same as previous results for SGD (or DP-SGD) in convex cases with delaying learning rates [25, 222]. Compared with DP-SGD, the added privacy noise would be reduced

from $\frac{1}{T}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\mathbb{E}[\|N^t\|^2]$ to $\frac{1}{T}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\mathbb{E}[\|N^t\|^2_{D^t}]$ when the gradients are sparse (so that $\|D^t\|_1 < d$ in adaptive iterations). Hence, this theorem suggests some constant improvements relative to DP-SGD when we switch for a constant number of times.

6.9.2 Non-Convex Cases

Assumption 7. *Stochastic gradient variance is bounded, i.e.,* $\mathbb{E}[\|g^{i,t} - \mathbb{E}[g^{i,t}]\|_2^2] \leqslant \tau^2$ *for all* i, t.

Theorem 22 (Convergence of Algorithm 11 for non-convex problems.)**.** *Let Assumptions 2,3,5, and 7 hold. Define constant* M *as* $M := C^2 + \frac{\sigma^2 C^2}{sb^2}$. *Under any delay parameter* s, *after running Algorithm 11 with constant learning rates* $\alpha^t = \alpha$ *such that* $\frac{L\alpha}{\epsilon} \leqslant 1$, *we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(w^t)\|^2] \leqslant \frac{2(\sqrt{M}+1)F(w^1)}{\alpha T} + 2\alpha L(\sqrt{M}+1)\left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right).$$

The proof is deferred to Appendix 6.12. Compared with Theorem 21, here we do not have constraints on s. Note that to guarantee (ε, δ)-DP by running T iterations, we can set $\sigma^2 = O\left(\frac{b^2 T\log(1/\delta)}{n^2\varepsilon^2}\right)$, $\alpha = O\left(\frac{1}{\sqrt{d}}\right)$, and $T = O\left(\frac{n\varepsilon}{\log(1/\delta)}\right)$, to arrive at a convergence bound $O\left(\frac{\sqrt{d}}{n\varepsilon} + \frac{\tau^2}{\sqrt{d}b}\right)$. Under any s, our rate (with and without noise) is the same as previous results on DP-SGD and (DP) adaptive methods for non-convex problems [187, 318]. We note that our non-convex analysis does not directly highlight the benefits of adaptivity or tradeoffs around s; hence the optimal choice of s according to this result is $s = T$, to maximize the goal of reducing privacy noise. However, the practical performance can be better than the upper bound derived here, as shown in our experiments (Section 6.10). Most of the previous works studying stochastic non-convex adaptive optimization does not prove improvements relative to SGD [e.g., 10, 63, 297, 318]. It is still an open problem to rigorously characterize the benefits of adaptivity for non-convex problems, which we leave for future work.

6.10 Experiments on DP^2

In this section we report empirical results on a range of learning tasks. In Section 6.10.1, we compare DP^2 with the baselines of DP-SGD and vanilla DP adaptive methods across various privacy budgets, and investigate the effects of delay on all datasets. We additionally compare DP^2 with recent more advanced private adaptive methods in Section 6.10.2, and conduct ablation studies to validate the effectiveness of different DP^2 components in Section 6.10.3.

In all experiments, we use Rényi differential privacy (RDP) accountant for the subsampled Gaussian mechanism [213] for privacy accounting. We focus on the RMSProp optimizer [117] and provide results relating to other adaptive methods such as AdaGrad [78, 273] in Appendix 6.13. Our experiments are implemented in JAX [40] with

Haiku [116] to auto-vectorize over the per-example operations (e.g. per-example clipping) for substantial speedups [275]. Unless explicitly stated, we report results with the best grid-searched hyperparameters. Note that for DP^2 we tune the learning rates and clipping thresholds separately for private SGD iterations and private adaptive (RMSProp) iterations. See Appendix 6.13.2 for hyperparameter details. Our code is publicly available at `github.com/kenziyuliu/DP2`.

Tuning s. In all experiments, we tune the delay parameter (s) via grid search. For convex tasks, we choose s from $\{0.025, 0.5, 0.1, 0.5, 1, 2\}$ epochs. For the non-convex model, we choose s from $\{0.5, 3, 10, 25\}$ epochs. We explore the sensitivity of DP^2 to s in Section 6.10.2, and show that there exist a wide range of s parameters that result in superior performance compared with baseline methods.

Datasets and Tasks. We pick datasets and tasks where adaptivity is crucial (e.g., those involving sparse gradients). For such tasks, adaptive methods have major benefits relative to SGD in non-private training, and we expect DP^2 to retain the benefits in private training. See Appendix 6.13.1 for a detailed description. For all datasets, we explore the effects of several noise multiplier (σ) values, and set $\delta = 10^{-k}$ where k is the smallest integer that satisfies $10^{-k} \leqslant 1/n$ for the training dataset size n.

6.10.1 DP^2 Compared with DP-SGD and Vanilla DP Adaptive Methods

We consider two popular baselines: DP-SGD [4] and vanilla DP-RMSProp [332]. In vanilla DP adaptive methods, private gradients are plugged into adaptive updating rules to approximate the preconditioners at each iteration. Figure 6.10 compares DP^2-RMSProp with DP-SGD and DP-RMSProp. We observe that across all datasets, DP^2 consistently and substantially outperforms the baselines in terms of both convergence and absolute performance.

Privacy/Utility tradeoffs. Figure 6.10 reports learning curves under specific privacy budgets determined by the batch size and the number of epochs. Here, we additionally explore privacy/utility tradeoffs across a range of privacy parameters, where ε ranges are consistent with prior works [e.g., 143]. Results are shown in Figure 6.11. We observe that similar to the results in Figure 6.10, DP^2 significantly outperforms DP-SGD and DP-RMSProp under each privacy budget. For reference, the non-private RMSProp method achieves 87% accuracy, 62% accuracy, and 0.88 mean square error (MSE) on IMDB, StackOverflow, and MovieLens, respectively. Indeed, with weaker privacy (larger ε), we expect smaller utility gaps between private and non-private optimization. In Appendix 6.13.4, we additionally explore how increasing the computational budget may affect the privacy-utility tradeoff.

Effects of s. Finally, we empirically study the effect of the delay parameter s. Intuitively, there exists a tradeoff between the amount of delay and the privacy noise in

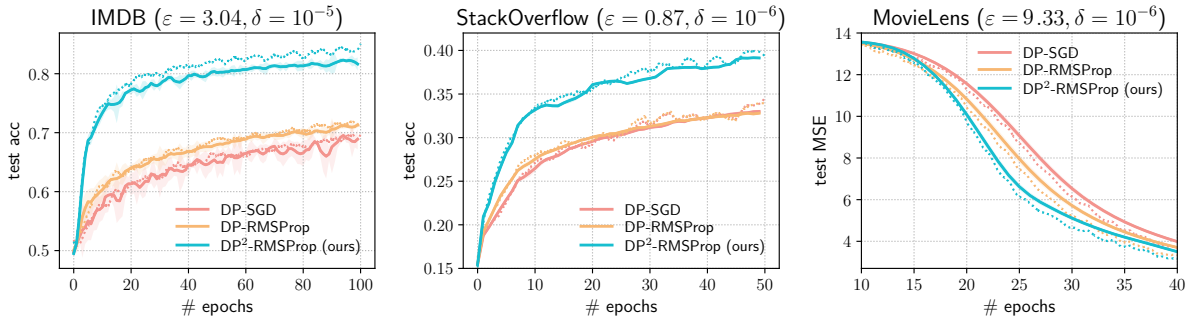


Figure 6.10: **Test performance** of DP^2 compared to DP-SGD and DP-RMSProp on IMDB (left), StackOverflow (middle), and MovieLens-100k (right) for a fixed privacy budget. For all datasets, we calculate the privacy loss (ε) under fixed δ's, noise multipliers {1.0, 1.0, 0.5}, and batch size 64. All runs are repeated over 5 random seeds. Dotted lines correspond to training metrics.

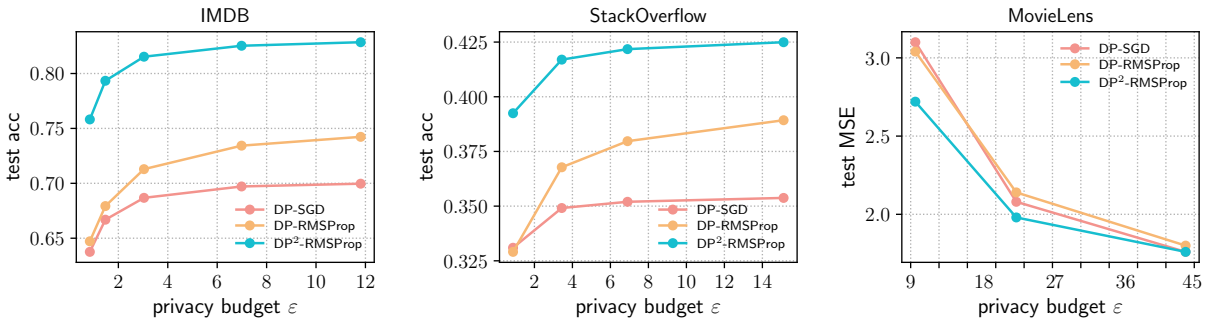


Figure 6.11: **Privacy/utility tradeoffs** of DP^2-RMSProp (Algorithm 11) compared with DP-SGD and DP-RMSProp for a range of privacy budgets. We see that DP^2-RMSProp consistently achieves more favorable privacy/utility tradeoffs than the baseline methods.

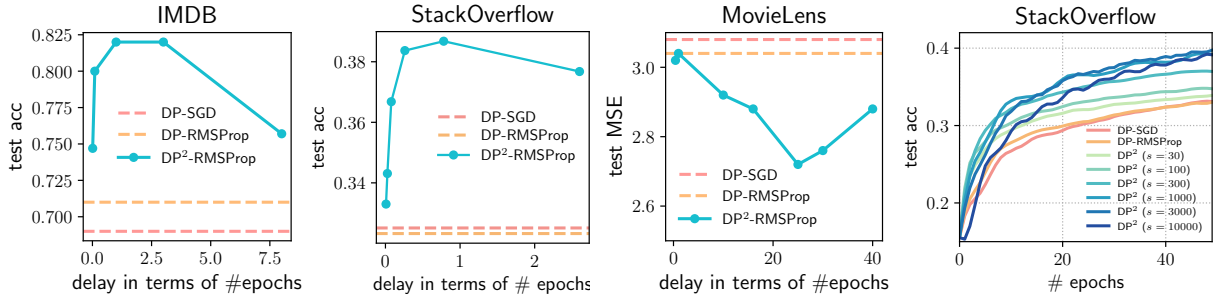


Figure 6.12: **Effect of the delay parameter** s. We show tradeoffs between delay and noise in the first three subplots. The rightmost subfigure showcases convergence curves under different delays (s=10000 corresponds to delaying for $\approx$ 3 epochs) where DP^2 achieves 4$\times$ convergence speedup than DP-SGD. Privacy settings follow those of Figure 6.10. Although a specific value of s achieves the greatest improvements, we observe that nearly all instantiations of DP^2 improve upon the baselines.

the preconditioner: averaging over more historical gradients (larger s) could yield less noisy preconditioners, while introducing more staleness. In Figure 6.12, we report test performance versus the delay s across all datasets on the first three subplots. In the last subplot, we additionally show the convergence behavior under different values of s. These results suggest that there is a "sweet spot" for s to yield good performance—small delays are gradually improving over DP-RMSProp; moderate delays perform best in terms of convergence and absolute performance; and large delays may slow down convergence (although it is possible to reach similar performance with sufficient training). These empirical results are consistent with the implications of our convergence analysis discussed in Section 6.9.1.

6.10.2 DP^2 Compared with Recent Methods for Private Optimization

As discussed in Chapter 2, beyond DP-SGD and vanilla DP adaptive methods, another line of work uses *auxiliary, public data* to improve private (adaptive) optimization. While *not directly comparable* to DP^2 since DP^2 does not require any side/public information, we compare DP^2 to two state-of-the-art methods along this direction[4]: (1) AdadPS [187] which uses public data or their statistics to estimate gradient geometry, and (2) PDA-DPMD [15], which uses the loss on public data as a mirror map to learn the underlying gradient geometry. Results are reported in Table 6.11, which show that DP^2 has comparable performance to state-of-the-art baselines, but without the need to access auxiliary data. See Appendix 6.13.6 for full details and convergence curves.

Dataset	**DP-SGD**	**DP-RMSProp**	**PDA-DPMD**	**AdaDPS** (w/ RMSProp)	**DP^2-RMSProp**
IMDB ↑	.687 ± .018	.713 ± .005	.703 ± .005	**.826** ± .003	**.815** ± .011
StackOverflow ↑	.330 ± .002	.328 ± .002	.353 ± .001	**.406** ± .027	**.391** ± .001
MovieLens ↓	3.02 ± .068	2.96 ± .062	3.74 ± .053	2.86 ± .042	**2.78** ± .054

Table 6.11: DP^2 compared with other private (adaptive) methods that use public data [15, 187]. Even though DP^2 *does not* require auxiliary information, we find that it achieves comparable performance with these state-of-the-art approaches that require additional public data. Corresponding convergence plots are presented in Figure 6.17 in the appendix.

6.10.3 Ablation Studies

[4]We do not directly compare with the prior work of Asi et al. [20] as the code is not publicly available and implementation details are missing in the paper; however, the more recent PDA-DPMD work of Amid et al. [15] we compare with suggests superior performance to Asi et al. [20]. We also implement the diagonal variant of the method proposed in the theoretically-focused work of Kairouz et al. [142], but observe that accuracy improves only marginally beyond random guessing (see Figure 6.18 in the appendix).

Finally, we also study the effectiveness of different components of DP^2. Recall that in Algorithm 11, we use noisy gradients from DP-SGD iterations to update both the model parameters and the preconditioner such that the total privacy cost is identical to that of DP-SGD. The first variant considers accumulating DP-SGD gradients in the same way, but it runs private adaptive methods using delayed preconditioner in almost all iterations. This requires us to add independent noise *twice* at most iterations (when accumulating the preconditioner and when noising the preconditioned update), thus increasing the total privacy budget. The second variant is identical to DP^2 except that it applies the delayed preconditioner *after* noising the clean gradient; this is to study the order of preconditioning as discussed in Section 6.8. As illustrated in Figure 6.13, both variants indeed significantly underperform our proposed method on the IMDB dataset, thus validating the design choices of DP^2. We defer complete results to Figure 6.16 and Table 6.14 in Appendix 6.13.5. See also Appendix 6.14 for the exact algorithms of both variants.

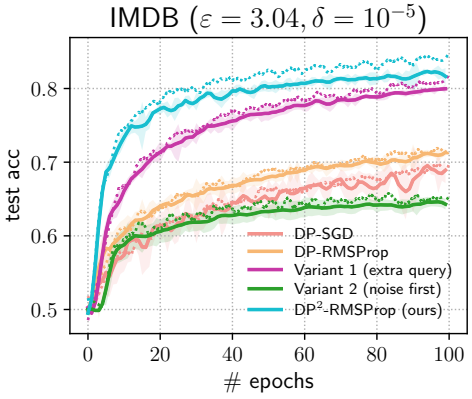


Figure 6.13: Different ablation variants of DP^2 on IMDB. The dotted lines correspond to training accuracy.

Conclusion and Future Work. In this section, we proposed DP^2, a private adaptive optimization framework that uses historical gradients to construct delayed but less noisy preconditioners, yielding improved privacy/utility tradeoffs *without the need to access auxiliary data*. We demonstrated the effectiveness of DP^2 both theoretically and empirically. In the future, it would be interesting to extend the techniques developed herein to other privacy-sensitive applications such as federated learning [206, 235]. It is also worth exploring interplays between DP^2 and private online optimization with tree aggregation, which similarly releases cumulative statistics with reduced noise [46].

6.11 Complete Proofs

Lemma 21. *Under Assumption 3, let $s_1 = s_2 = s$ in Algorithm 11, we have for any $j \in [d]$, $\mathbb{E}[v_j] \leqslant C^2 + \frac{\sigma^2 C^2}{sb^2}$.*

Proof. Recall that C is the gradient norm bound (Assumption 3). Let the clipping threshold be C as well. We have for $j \in [d]$,

$$\mathbb{E}\left[\left(\frac{1}{s}G_j\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{s}\left(g_j^{i_1} + \cdots + g_j^{i_s}\right) + \frac{1}{s}\left(N_j^{i_1} + \cdots + N_j^{i_s}\right)\right)^2\right] \tag{6.71}$$

$$= \mathbb{E}\left[\frac{1}{s^2}\left(g_j^{i_1} + \cdots + g_j^{i_s}\right)^2\right] + \mathbb{E}\left[\frac{1}{s^2}\left(N_j^{i_1} + \cdots + N_j^{i_s}\right)^2\right] \tag{6.72}$$

$$\leqslant C^2 + \frac{\sigma^2 C^2}{sb^2}, \tag{6.73}$$

where $\{i_1, \ldots, i_s\}$ denotes the indices of s noisy gradients used to obtain G_j, and $\{N_j^{i_1}, \ldots, N_j^{i_s}\}$ are random zero-mean Gaussian variables with variance $\frac{\sigma^2 C^2}{b^2}$ under noise multiplier σ, clipping threshold C, and mini-batch size b. Hence for any $j \in [d]$ and $t \in [T]$,

$$\mathbb{E}\left[\left(\frac{1}{s}G_j\right)^2\right] \leqslant C^2 + \frac{\sigma^2 C^2}{sb^2} := M, \tag{6.74}$$

$$\mathbb{E}[v_j] \leqslant M, \tag{6.75}$$

$$\mathbb{E}\left[\sqrt{v_j}\right] \leqslant \sqrt{\mathbb{E}[v_j]} \leqslant \sqrt{M} \tag{6.76}$$

$$\mathbb{E}\left[D_j^t\right] \leqslant \max\left\{\sqrt{M} + \epsilon, 1\right\}. \tag{6.77}$$

□

6.11.1 Proof of Theorem 21

Based on the updating rule, we have

$$\left\|w^{t+1} - w^*\right\|_{D^t}^2 \tag{6.78}$$

$$= \left\|w^t - \alpha^t \frac{g^t}{D^t} - \alpha^t N^t - w^*\right\|_{D^t}^2 \tag{6.79}$$

$$= \left\|w^t - w^*\right\|_{D^t}^2 + \left\|\alpha^t \frac{g^t}{D^t} + \alpha^t N^t\right\|_{D^t}^2 - 2\left\langle w^t - w^*, \alpha^t g^t + \alpha^t D^t N^t\right\rangle \tag{6.80}$$

$$= \left\|w^t - w^*\right\|_{D^t}^2 - 2\alpha^t\left\langle g^t, w^t - w^*\right\rangle + (\alpha^t)^2\left\langle g^t, \frac{g^t}{D^t}\right\rangle$$
$$- 2\alpha^t\langle w^t - w^*, D^t N^t\rangle + (\alpha^t)^2\|N^t\|_{D^t}^2 + 2(\alpha^t)^2\langle g^t, N^t\rangle. \tag{6.81}$$

Rearranging terms gives

$$\left\langle g^t, w^t - w^* \right\rangle = \frac{\|w^t - w^*\|_{D^t}^2 - \|w^{t+1} - w^*\|_{D^t}^2}{2\alpha^t} + \frac{\alpha^t}{2}\left\langle g^t, \frac{g^t}{D^t} \right\rangle - \langle w^t - w^*, D^t N^t \rangle + \frac{\alpha^t}{2}\|N^t\|_{D^t}^2 + \alpha^t \langle g^t, N^t \rangle. \tag{6.82}$$

Taking the expectation on both sides conditioned on w^t,

$$\left\langle \nabla F(w^t), w^t - w^* \right\rangle = \frac{\mathbb{E}_t\left[\|w^t - w^*\|_{D^t}^2\right] - \mathbb{E}_t[\|w^{t+1} - w^*\|_{D^t}^2]}{2\alpha^t} + \frac{\alpha^t}{2}\mathbb{E}_t\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle\right] + \frac{\alpha^t}{2}\mathbb{E}_t\left[\|N^t\|_{D^t}^2\right], \tag{6.83}$$

where we have used the fact that N is a zero-mean Gaussian variable independent of g^t, w^t. Taking the expectation on both sides and using the convexity of $F(\cdot)$:

$$\begin{aligned} &\mathbb{E}[F(w^t)] - F(w^*) \\ &\leqslant \frac{\mathbb{E}[\|w^t - w^*\|_{D^t}^2] - \mathbb{E}[\|w^{t+1} - w^*\|_{D^t}^2]}{2\alpha^t} + \frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle\right] + \frac{\alpha^t}{2}\mathbb{E}\left[\|N^t\|_{D^t}^2\right]. \end{aligned} \tag{6.84}$$

Applying telescope sum, we have

$$\begin{aligned} &\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)] - F(w^*)\right) \\ &\leqslant \frac{\|w^1 - w^*\|_{A^1}^2}{2\alpha_1} + \sum_{t=2}^{T}\left(\frac{\mathbb{E}\left[\|w^t - w^*\|_{D^t}^2\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|w^t - w^*\|_{D^{t-1}}^2\right]}{2\alpha_{t-1}}\right) \\ &\quad + \sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle\right] + \sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\|N^t\|_{D^t}^2\right]. \end{aligned} \tag{6.85}$$

Hence, we need to bound the RHS:

$$\begin{aligned} &\frac{\|w^1 - w^*\|_{D^1}^2}{2\alpha^2} + \underbrace{\sum_{t=2}^{T}\left(\frac{\mathbb{E}\left[\|w^t - w^*\|_{D^t}^2\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|w^t - w^*\|_{D^{t-1}}^2\right]}{2\alpha^{t-1}}\right)}_{T_1} \\ &\quad + \underbrace{\sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle\right]}_{T_2} + \sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\|N^t\|_{D^t}^2\right], \end{aligned} \tag{6.86}$$

where the vector $D^t \in \mathbb{R}^d$ satisfies that $D^t = \mathbf{1}$ when running private SGD steps, and $D^t = \sqrt{v} + \epsilon$ when running private RMSProp steps.

Let the delay parameter to be scheduled as

$$s = vT \ (0 < v < 1) \tag{6.87}$$

and the learning rate α^t be

$$\alpha^t \leftarrow \frac{\alpha^{\left\lfloor \frac{t}{2s} \right\rfloor + \left\lfloor \frac{t+s}{2s} \right\rfloor + 1}}{\sqrt{t}}, \tag{6.88}$$

where $\alpha = \min\left\{\epsilon, \frac{1}{\sqrt{M+\epsilon}}, 1\right\}$, and M is the upper bound of $\mathbb{E}\left[v_j\right]$ for $j \in [d]$, as defined and proved in Lemma 21.

We next consider the T_1 term. There are four cases.

1. **DP-SGD at the $t-1$-th iteration, and DP-SGD at the t-th iteration**: As $D^t = D^{t-1}$ there is not much requirement other that the learning rates need to satisfy $\alpha^t \leqslant \alpha^{t-1}$, which holds for our choice.
2. **Private RMSProp at the $t-1$-th iteration, and private RMSProp at the t-th iteration**: Similar to previous case, the learning rates need to satisfy $\alpha^t \leqslant \alpha^{t-1}$, which holds for our choice.
3. **DP-SGD at the $t-1$-th iteration, and private RMSProp at the t-th iteration**: We require

$$\frac{\alpha^t}{\epsilon} \leqslant \alpha^{t-1} \implies \frac{\sqrt{v^t} + \epsilon}{\alpha^t} \geqslant \frac{1}{\alpha^{t-1}} \tag{6.89}$$

   But in this case we must have $t \,\%\, s = 0$. So this is satisfied by our choice as long as $\alpha \leqslant \epsilon$.
4. **Private RMSProp at the $t-1$-th iteration, and DP-SGD at the t-th iteration**

The first three cases form an updating pattern of DP-SGD $\rightarrow \cdots \rightarrow$ DP-SGD $\rightarrow$ DP-RMSProp$\rightarrow \cdots \rightarrow$ DP-RMSProp, where every pattern takes $2s$ iterations, except for the first pattern, because the telescope sum starts from $t = 2$. For the first pattern, we have

$$\frac{\left\|w^1 - w^*\right\|_{D^1}^2}{2\alpha^2} + \sum_{t=2}^{2s}\left(\frac{\mathbb{E}\left[\|w^t - w^*\|_{D^t}^2\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|w^t - w^*\|_{D^{t-1}}^2\right]}{2\alpha^{t-1}}\right) \tag{6.90}$$

$$= \frac{\left\|w^1 - w^*\right\|_{D^1}^2}{2\alpha^2} + \sum_{t=2}^{2s}\left(\mathbb{E}\left[\left\|w^t - w^*\right\|_{\frac{D^t}{\alpha^t} - \frac{D^{t-1}}{\alpha^{t-1}}}^2\right]\right) \tag{6.91}$$

$$\leqslant \frac{\left\|w^1 - w^*\right\|_{D^1}^2}{2\alpha^2} + R^2\sum_{t=2}^{2s}\left(\frac{\mathbb{E}\left[\|D^t\|_1\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|D^{t-1}\|_1\right]}{2\alpha^{t-1}}\right) \leqslant \frac{R^2}{2\alpha^{2s}}\mathbb{E}\left[\|D^{2s}\|_1\right], \tag{6.92}$$

where $D^{2s} = \sqrt{v} + \epsilon$.

For $k \geqslant 1$, we have

$$\begin{aligned}
&\sum_{t=2sk+1}^{2sk+2s} \left(\frac{\mathbb{E}\left[\|w^t - w^*\|^2_{D^t}\right]}{2\alpha^t} - \frac{\mathbb{E}\left[\|w^t - w^*\|^2_{D^{t-1}}\right]}{2\alpha^{t-1}} \right) \\
&= \frac{\mathbb{E}\left[\|w^{2sk+1} - w^*\|^2_{D^{2sk+1}}\right]}{2\alpha^{2sk+1}} - \frac{\mathbb{E}\left[\|w^{2sk+1} - w^*\|^2_{D^{2sk}}\right]}{2\alpha^{2sk}} + \sum_{t=2sk+2}^{2sk+2s} \left(\mathbb{E}\left[\left\|w^t - w^*\right\|^2_{\frac{D^t}{2\alpha^t} - \frac{D^{t-1}}{2\alpha^{t-1}}} \right] \right) \\
&\leqslant \frac{\mathbb{E}\left[\|w^{2sk+1} - w^*\|^2_{D^{2sk+1}}\right]}{2\alpha^{2sk+1}} - \frac{\mathbb{E}\left[\|w^{2sk+1} - w^*\|^2_{D^{2sk}}\right]}{2\alpha^{2sk}} + R^2 \left(\frac{\mathbb{E}[\|D^{2sk+2s}\|_1]}{2\alpha^{2sk+2s}} - \frac{\mathbb{E}[\|D^{2sk+1}\|_1]}{2\alpha^{2sk+1}} \right) \\
&\leqslant \frac{\mathbb{E}\left[\|w^{2sk+1} - w^*\|^2_{D^{2sk+1}}\right]}{2\alpha^{2sk+1}} + R^2 \left(\frac{\mathbb{E}[\|D^{2sk+2s}\|_1]}{2\alpha^{2sk+2s}} - \frac{\mathbb{E}[\|D^{2sk+1}\|_1]}{2\alpha^{2sk+1}} \right) \\
&\leqslant \frac{R^2}{2\alpha^{2sk+2s}} \mathbb{E}\left[\|D^{2sk+2s}\|_1\right], \qquad (6.93)
\end{aligned}$$

where $D^{2sk+2s} = \sqrt{v} + \epsilon$ belong to DP-RMSProp updates.

We look at the second T_2 term, and prove by induction that there exists a constant κ such that

$$\sum_{t=1}^{T} \frac{\alpha^t}{2} \mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle \right] \leqslant \frac{\kappa}{\alpha^T} \mathbb{E}\left[\|D^T\|_1\right]. \qquad (6.94)$$

When $T = 1$ ($\alpha^1 = \alpha$ and $D^1 = \mathbf{1}$), $\frac{\alpha}{2}\mathbb{E}[\|g^1\|^2] \leqslant \frac{\kappa d}{\alpha}$ holds if $\kappa \geqslant \alpha^2 C^2$. At each step t, the goal is to get

$$\frac{\kappa}{\alpha^{t-1}} \mathbb{E}\left[\|D^{t-1}\|_1\right] + \frac{\alpha^t}{2} \mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t} \right\rangle \right] \leqslant \frac{\kappa}{\alpha^t} \mathbb{E}\left[\|D^t\|_1\right] \qquad (6.95)$$

1. **DP-SGD at the $t-1$-th iteration, and DP-SGD at the t-th iteration**: We require

$$\frac{\kappa d}{\alpha^{t-1}} + \frac{\alpha^t}{2} \mathbb{E}\left[\left\|g^t\right\|^2\right] \leqslant \frac{\kappa d}{\alpha^t} \qquad (6.96)$$

   which would hold for choice of α^t as gradients are bounded and $\kappa \geqslant \alpha^2 C^2$.

2. **Private RMSProp at the $t-1$-th iteration, and private RMSProp at the t-th iteration**: We need

$$\frac{\kappa \mathbb{E}\left[\|\sqrt{v^{t-1}} + \epsilon\|_1\right]}{\alpha^{t-1}} + \frac{\alpha^t}{2} \mathbb{E}\left[\left\langle g^t, \frac{g^t}{\sqrt{v^{t-1}} + \epsilon} \right\rangle \right] \leqslant \frac{\kappa}{\alpha^t} \mathbb{E}\left[\|\sqrt{v^t} + \epsilon\|_1\right], \qquad (6.97)$$

$$\frac{\alpha^t}{2} \mathbb{E}\left[\left\langle g^t, \frac{g^t}{\sqrt{v^{t-1}} + \epsilon} \right\rangle \right] \leqslant \left(\frac{\kappa}{\alpha^t} - \frac{\kappa}{\alpha^{t-1}} \right) \mathbb{E}\left[\left\| \sqrt{v^{t-1}} + \epsilon \right\|_1 \right]. \qquad (6.98)$$

Let

$$h(s) \geqslant \max_{t\in[T]} \left\{ \frac{\mathbb{E}\left[\|g^t\|_1\right]}{\mathbb{E}\left[\left\|\frac{1}{s}\left|G^{\lfloor \frac{t}{s} \rfloor s}\right| + \epsilon\right\|_1\right]} \right\}. \tag{6.99}$$

Based on our updating rule,

$$\mathbb{E}\left[\left\|\sqrt{v^t} + \epsilon\right\|_1\right] \geqslant \sqrt{1-\beta}\, \mathbb{E}\left[\left\|\frac{1}{s}\left|G^{\lfloor \frac{t}{s} \rfloor s}\right| + \epsilon\right\|_1\right]. \tag{6.100}$$

Note that

$$\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{\sqrt{v^{t-1}}+\epsilon}\right\rangle\right] \leqslant \frac{\alpha^t}{2}\mathbb{E}\left[\frac{\|g^t\|^2}{\epsilon}\right] \leqslant \frac{\alpha^t C}{2\epsilon}\mathbb{E}[\|g^t\|] \leqslant \frac{\alpha^t C}{2\epsilon}\mathbb{E}[\|g^t\|_1], \tag{6.101}$$

where we have used the assumption that $\|g^t\| \leqslant C$. Combining the above two,

$$\begin{aligned}
\frac{\alpha^t C}{2\epsilon}\mathbb{E}[\|g^t\|] &\leqslant \frac{\alpha^t C}{2\epsilon} h(s)\mathbb{E}\left[\left\|\frac{1}{s}\left|G^{\lfloor \frac{t}{s} \rfloor s}\right| + \epsilon\right\|_1\right] && (6.102)\\
&\leqslant \frac{\alpha^t C}{2\epsilon}\frac{h(s)}{\sqrt{1-\beta}}\mathbb{E}\left[\left\|\sqrt{v^{t-1}} + \epsilon\right\|_1\right] && (6.103)\\
&\leqslant \kappa\left(\frac{1}{\alpha^t} - \frac{1}{\alpha^{t-1}}\right)\mathbb{E}\left[\left\|\sqrt{v^{t-1}} + \epsilon\right\|_1\right]. && (6.104)
\end{aligned}$$

This implies the condition holds as long as κ satisfies

$$\kappa \geqslant \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}. \tag{6.105}$$

3. **DP-SGD at the $t-1$-th iteration, and private RMSProp at the t-th iteration.** We want to prove

$$\frac{\kappa d}{\alpha^{t-1}} + \frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{D^t}\right\rangle\right] \leqslant \frac{\kappa}{\alpha^t}\mathbb{E}\left[\left\|D^t\right\|_1\right]. \tag{6.106}$$

As $\|g^t\| \leqslant C$, it holds that

$$\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{\sqrt{v^t}+\epsilon}\right\rangle\right] \leqslant \frac{\alpha^t}{2\epsilon}\mathbb{E}[\|g^t\|^2] \leqslant \frac{\alpha^t C}{2\epsilon}\mathbb{E}[\|g^t\|] \leqslant \frac{\alpha^t C}{2\epsilon}\mathbb{E}[\|g^t\|_1]. \tag{6.107}$$

Therefore,

$$\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t, \frac{g^t}{\sqrt{v^t}+\epsilon}\right\rangle\right] \leqslant \frac{Ch(s)}{2\epsilon\sqrt{1-\beta}}\alpha^t\mathbb{E}\left[\left\|\sqrt{v^t} + \epsilon\right\|_1\right]. \tag{6.108}$$

Based on our learning rate set in Eq. (6.88),

$$\sqrt{t}\alpha^t = \sqrt{t-1}\alpha^{t-1}\epsilon \tag{6.109}$$

$$\implies \frac{\alpha^t}{2} \leqslant \frac{1}{\alpha^t} - \frac{1}{\alpha^{t-1}\epsilon} \leqslant \frac{1}{\alpha^t} - \frac{d}{\alpha^{t-1}\mathbb{E}\left[\|D^t\|_1\right]}. \tag{6.110}$$

Hence,

$$\frac{Ch(s)}{2\epsilon\sqrt{1-\beta}}\alpha^t\mathbb{E}\left[\left\|\sqrt{v^t}+\epsilon\right\|_1\right] \leqslant \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}\mathbb{E}\left[\|D^t\|_1\right]\left(\frac{1}{\alpha^t} - \frac{d}{\alpha^{t-1}\mathbb{E}\left[\|D^t\|_1\right]}\right) \tag{6.111}$$

$$\leqslant \kappa\left(\frac{\mathbb{E}[\|D^t\|_1]}{\alpha^t} - \frac{d}{\alpha^{t-1}}\right), \tag{6.112}$$

where we require

$$\kappa \geqslant \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}. \tag{6.113}$$

4. **Private RMSProp at the $t-1$-th iteration, and DP-SGD at the t-th iteration.** We need

$$\frac{\kappa}{\alpha^{t-1}}\mathbb{E}\left[\left\|\sqrt{v^{t-1}}+\epsilon^{t-1}\right\|_1\right] + \frac{\alpha^t}{2}\mathbb{E}\left[\|g^t\|^2\right] \leqslant \frac{\kappa d}{\alpha^t}. \tag{6.114}$$

Plug in $\mathbb{E}\left[\|\sqrt{v^{t-1}}\|_1\right] \leqslant d\sqrt{M}$ (Lemma 21) and $\|g^t\|^2 \leqslant C^2$, we have

$$\frac{\kappa}{\alpha^{t-1}}\mathbb{E}\left[\|\sqrt{v^{t-1}}+\epsilon\|_1\right] + \frac{\alpha^t}{2}\mathbb{E}\left[\|g^t\|^2\right] \leqslant \frac{\kappa}{\alpha^{t-1}}\left(d\sqrt{M}+d\right) + \frac{\alpha^t}{2}C^2. \tag{6.115}$$

Based on our learning rate set in Eq. (6.88), for some constant γ,

$$\alpha^{t-1} = \frac{\gamma}{\sqrt{t-1}},\ \alpha^t \leqslant \frac{\gamma}{\sqrt{t}(\sqrt{M}+1)} \tag{6.116}$$

$$\implies \frac{\alpha^t}{2} \leqslant \frac{1}{\alpha^t} - \frac{\sqrt{M}+1}{\alpha^{t-1}} \leqslant \frac{d}{\alpha^t} - \frac{d\sqrt{M}+d}{\alpha^{t-1}}. \tag{6.117}$$

Therefore

$$\frac{\alpha^t}{2}C^2 \leqslant \kappa\left(\frac{d}{\alpha^t} - \frac{d\sqrt{M}+d}{\alpha^{t-1}}\right) \tag{6.118}$$

holds as long as $\kappa \geqslant \alpha^2C^2$. To sum up, the requirement on κ is

$$\kappa \geqslant \max\left\{\alpha^2C^2, \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}\right\}. \tag{6.119}$$

Final convergence results:

$$\min_{t\in[T]} \mathbb{E}\left[F(w^t)\right] - F(w^*) \tag{6.120}$$

$$\leqslant \frac{R^2+\kappa}{\alpha\left\lfloor\frac{1}{2v}\right\rfloor+\left\lfloor\frac{1+v}{2v}\right\rfloor}\frac{1}{\sqrt{T}}\sum_{t\in T_v}\mathbb{E}\left[\left\|D^t\right\|_1\right] + \frac{1}{T}\sum_{t=1}^{T}\frac{\alpha^{\left\lfloor\frac{t}{2vT}\right\rfloor+\left\lfloor\frac{t+vT}{2vT}\right\rfloor}}{\sqrt{t}}\mathbb{E}[\|N^t\|_{D^t}^2], \tag{6.121}$$

where T_v denotes the iteration indices where we switch from private RMSProp steps to private SGD steps plus the last iteration, and its cardinality is $|T_v| = \lceil\frac{1}{2v}\rceil$, and $\kappa \geqslant \max\left\{\alpha^2C^2, \frac{Ch(s)}{\epsilon\sqrt{1-\beta}}, \right\}$, $\alpha = \min\left\{\epsilon, \frac{1}{\sqrt{M}+\epsilon}, 1\right\}$.

6.11.2 A Closer Look at $h(s)$

We closely examine $h(s)$, defined as

$$h(s) \geqslant \max_{t\in[T]}\left\{\frac{\mathbb{E}\left[\|g^t\|_1\right]}{\mathbb{E}\left[\left\|\frac{1}{s}\left|G^{\lfloor\frac{t}{s}\rfloor s}\right| + \epsilon\right\|_1\right]}\right\}. \tag{6.122}$$

Let us assume mini-batch gradients on consecutive time steps are not very different, i.e. $\|g^t - g^{t-1}\|_1 \leqslant M$. This means each gradient norm cannot be too far away from each other, which can be used to show the dependence of $h(s)$ on the delay parameter s. Denote the gap between the current iteration t and the iteration where v gets updated as k, i.e., $k := t - \lfloor\frac{t}{s}\rfloor s$. Hence,

$$\frac{\left\|g^t\right\|_1}{\left\|\frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)+\frac{1}{s}\left(N^{t-k-1}+\cdots+N^{t-k-s}\right)\right\|_1 + d\epsilon} \tag{6.123}$$

$$= \frac{\left\|g^t - \frac{1}{s}\left(g^{t-k-1}+\cdots+g_j^{t-k-s}\right)+\frac{1}{s}\left(g^{t-k-1}+\cdots+g_j^{t-k-s}\right)\right\|_1}{\left\|\frac{1}{s}\left(g^{t-k-1}+\cdots+g_j^{t-k-s}\right)+\frac{1}{s}\left(N^{t-k-1}+\cdots+N^{t-k-s}\right)\right\|_1 + d\epsilon} \tag{6.124}$$

$$= \frac{\left\|\frac{1}{s}\left((g^t-g^{t-k-1})+\cdots+(g^t-g^{t-k-s})\right)+\frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)\right\|_1}{\left\|\frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)+\frac{1}{s}\left(N^{t-k-1}+\cdots+N^{t-k-s}\right)\right\|_1 + d\epsilon} \tag{6.125}$$

$$\leqslant \frac{\left\|\frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)\right\|_1}{\left\|\frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)+\frac{1}{s}\left(N^{t-k-1}+\cdots+N^{t-k-s}\right)\right\|_1 + d\epsilon} + \frac{\frac{1}{s}(sM+\cdots+(2s)M)}{d\epsilon} \tag{6.126}$$

Denote $a := \frac{1}{s}\left(N^{t-k-1}+\cdots+N^{t-k-s}\right)$, and $b := \frac{1}{s}\left(g^{t-k-1}+\cdots+g^{t-k-s}\right)$. Then

$$h(s) \leqslant \frac{\mathbb{E}[\|b\|_1]}{\mathbb{E}[\|a+b\|_1]+d\epsilon} + \frac{sM}{d\epsilon} \tag{6.127}$$

$$\leqslant \frac{1}{\left|\frac{\mathbb{E}[\|a\|_1]}{\mathbb{E}[\|b\|_1]} - 1\right| + \frac{d\epsilon}{\mathbb{E}[\|b\|_1]}} + \frac{sM}{d\epsilon} \tag{6.128}$$

In the special case where gradients are sparse, i.e., $\mathbb{E}[\|b\|_1] < \mathbb{E}[\|a\|_1]$, we have

$$h(s) \leqslant \frac{1}{\frac{\mathbb{E}[\|a\|_1]}{\mathbb{E}[\|b\|_1]} + \frac{d\epsilon}{\mathbb{E}[\|b\|_1]} - 1} + \frac{sM}{d\epsilon} \tag{6.129}$$

It is easy to see that the RHS is $O(s)$, and it increases as s. We can informally express it as $c_1 s + c_2$, where c_1 and c_2 are two constants.

6.12 Proof of Theorem 22

First we introduce a result that will be used in this section. Under the bounded stochastic gradient variance assumption (Assumption 7), we have that conditioned on w^t,

$$\mathbb{E}_t\left[\|g^t\|^2\right] \leqslant \frac{\tau^2}{b} + \|\nabla F(w^t)\|^2, \tag{6.130}$$

where b refers to the mini-batch size to obtain gradient g^t, i.e., $g^t \leftarrow \frac{1}{b}\sum_{i\in B} g^{i,t}$. This lemma is proved in Zaheer et al. [318]. The per-coordinate version of this result is that for $j \in [d]$,

$$\mathbb{E}_t\left[(g_j^t)^2\right] \leqslant \frac{\tau_j^2}{b} + \left(\nabla_j F(w^t)\right)^2, \tag{6.131}$$

and $\sum_{j\in[d]} \tau_j^2 = \tau^2$.

As we assume $F(w)$ is L-smooth, at each iteration t,

$$F(w^{t+1}) \leqslant F(w^t) + \langle \nabla F(w^t), w^{t+1} - w^t\rangle + \frac{L}{2}\left\|w^{t+1} - w^t\right\|^2. \tag{6.132}$$

Based on the updating rule of Algorithm 11, we have

$$F(w^{t+1}) \leqslant F(w^t) + \langle \nabla F(w^t), w^{t+1} - w^t\rangle + \frac{L}{2}\left\|w^{t+1} - w^t\right\|^2 \tag{6.133}$$

$$= F(w^t) - \alpha^t \left\langle \nabla F(w^t), \frac{g^t}{D^t} + N^t \right\rangle + \frac{(\alpha^t)^2 L}{2}\left\|\frac{g^t}{D^t} + N^t\right\|^2, \tag{6.134}$$

where $N \in \mathbb{R}^d$ and $N_j \sim \mathcal{N}\left(0, \frac{\sigma^2 C^2}{b^2}\right)$ with noise multiplier σ and clipping threshold C, and D^t satisfies that

$$D^t \leftarrow \begin{cases} \mathbf{1} & \text{if } t \bmod 2s \leqslant s, \\ \sqrt{v} + \epsilon & \text{otherwise.} \end{cases} \tag{6.135}$$

Take expectation with respect to samples at the t-th iteration and N^t,

$$\begin{aligned}\mathbb{E}_t[F(w^{t+1})] &\leqslant F(w^t) - \alpha^t \left\langle \nabla F(w^t), \frac{\nabla F(w^t)}{D^t} \right\rangle + \frac{(\alpha^t)^2 L}{2} \mathbb{E}_t \left[\left\| \frac{g^t}{D^t} \right\|^2 \right] + \frac{d(\alpha^t)^2 L}{2b^2} \sigma^2 C^2 \\ &= F(w^t) - \alpha^t \sum_{j \in [d]} \frac{(\nabla_j F(w^t))^2}{D_j^t} + \frac{(\alpha^t)^2 L}{2} \sum_{j \in [d]} \frac{\mathbb{E}_t \left[(g_j^t)^2 \right]}{(D_j^t)^2} + \frac{d(\alpha^t)^2 L}{2b^2} \sigma^2 C^2,\end{aligned} \tag{6.136}$$

where we have used the fact that N^t is a zero-mean random variable independent of w^t, and D^t is independent of samples at time t. We need to consider two cases.

1. **DP-SGD at the t-th iteration**

   In this case, $D^t = \mathbf{1}$. Hence plugging in

   $$\mathbb{E}_t \left[(g_j^t)^2 \right] \leqslant \frac{\tau_j^2}{b} + \left(\nabla_j F(w^t) \right)^2, \tag{6.137}$$

   we have

   $$\mathbb{E}_t \left[F(w^{t+1}) \right] \leqslant F(w^t) - \left(\alpha^t - \frac{(\alpha^t)^2 L}{2} \right) \left\| \nabla F(w^t) \right\|^2 + (\alpha^t)^2 L \left(\frac{\tau^2}{2b} + \frac{\sigma^2 C^2 d}{2b^2} \right). \tag{6.138}$$

   Under constant learning rate, let $\alpha^t = \alpha \leqslant \frac{1}{L}$,

   $$\mathbb{E}_t \left[F(w^{t+1}) \right] \leqslant F(w^t) - \frac{\alpha}{2} \|\nabla F(w^t)\|^2 + (\alpha^t)^2 L \left(\frac{\tau^2}{2b} + \frac{\sigma^2 C^2 d}{2b^2} \right). \tag{6.139}$$

   Taking expectation on both sides gives

   $$\frac{\alpha}{2} \mathbb{E} \left[\|\nabla F(w^t)\|_2^2 \right] \leqslant \mathbb{E}[F(w^t)] - \mathbb{E}[F(w^{t+1})] + (\alpha^t)^2 L \left(\frac{\tau^2}{2b} + \frac{\sigma^2 C^2 d}{2b^2} \right). \tag{6.140}$$

2. **Private RMSProp at the t-th iteration**

   We have

   $$\mathbb{E}_t[F(w^{t+1})] \leqslant F(w^t) - \alpha^t \sum_{j \in [d]} \frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t} + \epsilon} + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j \in [d]} \frac{\mathbb{E}_t[(g_j^t)^2]}{\sqrt{v_j^t} + \epsilon^t} + \frac{d(\alpha^t)^2 L \sigma^2 C^2}{2b^2}. \tag{6.141}$$

   Plugging in $\mathbb{E}_t \left[(g_j^t)^2 \right] \leqslant \frac{\tau_j^2}{b} + \left(\nabla_j F(w^t) \right)^2$ results in

   $\mathbb{E}_t[F(w^{t+1})]$

$$
\begin{aligned}
&\leqslant F(w^t) - \alpha^t \sum_{j\in[d]} \frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon} + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j\in[d]} \frac{\sigma_j^2}{\left(\sqrt{v_j^t}+\epsilon\right) b} \\
&\quad + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j\in[d]} \frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon} + \frac{d(\alpha^t)^2 L\sigma^2 C^2}{2b^2} \\
&= F(w^t) - \left(\alpha^t - \frac{(\alpha^t)^2 L}{2\epsilon}\right) \sum_{j\in[d]} \frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon} + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j\in[d]} \frac{\tau_j^2}{\left(\sqrt{v_j^t}+\epsilon\right) b} + \frac{d(\alpha^t)^2 L\sigma^2 C^2}{2b^2} \\
&\leqslant F(w^t) - \left(\alpha^t - \frac{(\alpha^t)^2 L}{2\epsilon}\right) \sum_{j\in[d]} \frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon} + (\alpha^t)^2 L \left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right).
\end{aligned}
$$

Taking expectation on both sides yields

$$
\mathbb{E}[F(w^{t+1})] \leqslant \mathbb{E}[F(w^t)] - \left(\alpha^t - \frac{(\alpha^t)^2 L}{2\epsilon}\right) \sum_{j\in[d]} \mathbb{E}\left[\frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon}\right] + (\alpha^t)^2 L \left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right).
$$

We need to lower bound $\sum_{j\in[d]} \mathbb{E}\left[\frac{[\nabla F(w^t)]_j^2}{\sqrt{v_j^t}+\epsilon}\right]$. We know from Holder's inequality that $\mathbb{E}[\langle u, v\rangle] \leqslant \mathbb{E}[\|u\|_1]\mathbb{E}[\|v\|_\infty]$. Now note that

$$
\begin{aligned}
\mathbb{E}\left[\|\nabla F(w^t)\|^2\right] = \mathbb{E}\left[\left\langle \frac{|\nabla F(w^t)|^2}{D^t}, D^t \right\rangle\right] &\leqslant \mathbb{E}\left[\left\| \frac{(\nabla F(w^t))^2}{D^t} \right\|_1\right] \mathbb{E}\left[\|D^t\|_\infty\right] \qquad (6.142) \\
&\leqslant \mathbb{E}\left[\left\| \frac{(\nabla F(w^t))^2}{D^t} \right\|_1\right] (\sqrt{M}+\epsilon). \qquad (6.143)
\end{aligned}
$$

Hence

$$
\sum_{j\in[d]} \mathbb{E}\left[\frac{(\nabla_j F(w^t))^2}{D_j^t}\right] \geqslant \frac{\mathbb{E}[\|\nabla F(w^t)\|^2]}{\sqrt{M}+\epsilon} \qquad (6.144)
$$

and

$$
\begin{aligned}
&\mathbb{E}[F(w^{t+1})] \\
&\leqslant \mathbb{E}[F(w^t)] - \left(\alpha^t - \frac{(\alpha^t)^2 L}{2\epsilon}\right) \frac{\mathbb{E}\left[\|\nabla F(w^t)\|^2\right]}{\sqrt{M}+\epsilon} + (\alpha^t)^2 L \left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right). \qquad (6.145)
\end{aligned}
$$

Let $\alpha^t = \alpha \leqslant \frac{\epsilon}{L}$, we obtain

$$
\begin{aligned}
&\mathbb{E}[F(w^{t+1})] \\
&\leqslant \mathbb{E}[F(w^t)] - \frac{\alpha}{2(\sqrt{M}+\epsilon)} \mathbb{E}\left[\|\nabla F(w^t)\|^2\right] + (\alpha^t)^2 L \left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right). \qquad (6.146)
\end{aligned}
$$

Combining the two cases, for any t, we have

$$\begin{aligned}&\mathbb{E}[\|\nabla F(w^t)\|^2] \\ &\leqslant \frac{2(\sqrt{M}+1)}{\alpha}\left(\mathbb{E}[F(w^t)] - \mathbb{E}[F(w^{t+1})]\right) + 2\alpha L(\sqrt{M}+1)\left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right). \end{aligned} \tag{6.147}$$

Taking a telescope sum results in

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(w^t)\|^2] \leqslant \frac{2(\sqrt{M}+1)F(w^1)}{\alpha T} + 2\alpha L(\sqrt{M}+1)\left(\frac{\tau^2}{2\epsilon^2 b} + \frac{d\sigma^2 C^2}{2b^2}\right), \tag{6.148}$$

where $M := C^2 + \frac{\sigma^2 C^2}{sb^2}$.

Algorithm 11 DP^2-RMSprop: Delayed Preconditioners for Differentially Private RMSprop

Input: T, batch size b, noise multiplier σ, clipping thresholds C, initial model $w^0 \in \mathbb{R}^d$, $v = \mathbf{0}$, constant $\epsilon \in \mathbb{R}_+$, learning rate schedule α^t, moving average parameter β, SGD cumulative aggregation step s_1, RMSProp cumulative step s_2
for $t = 0, \cdots, T-1$ **do**
  if $t \bmod (s_1 + s_2) = 0$ **then**
    Reset accumulator $G^t \leftarrow \mathbf{0}$
  end if
  if $t \bmod (s_1 + s_2) = s_1$ **then**
    Update moment estimates as $v \leftarrow \beta v + (1-\beta)\left(G^t/s_1\right)^2$
    Reset accumulator $G^t \leftarrow \mathbf{0}$
  end if
  Uniformly randomly sample a mini-batch B with size b from private training data
  Get individual gradients for sample $i \in B$: $g^{i,t} \leftarrow \nabla f(x^i; w^t)$
  Privatize the (preconditioned) gradients using the Gaussian mechanism:

$$\tilde{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(\frac{g^{i,t}}{D^t}, C\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C^2\right)\right)$$

  where

$$D^t \leftarrow \begin{cases} \mathbf{1} & \text{if } t \bmod (s_1 + s_2) < s_1 \\ \sqrt{v} + \epsilon & \text{otherwise.} \end{cases}$$

  Accumulate the private gradients $\tilde{g}^t$: $G^{t+1} \leftarrow G^t + \tilde{g}^t$
  Update model parameters w:

$$w^{t+1} \leftarrow w^t - \alpha^t \tilde{g}^t$$

end for
return w^T

6.13 Experimental Details and Additional Results

6.13.1 Datasets

- **IMDB** [199] is a binary classification dataset on sentiment analysis for movie reviews that includes 25,000/25,000 training/test samples. Each sample is a review under a vocabulary size of 10,000. We train a logistic regression model with 10,001 parameters.
- **StackOverflow** [21, 140] is a large-scale text dataset containing questions and answers from Stack Overflow. We focus on the task of classifying the tag(s) of a given sentence described in [21], though we focus on the usual centralized training setting instead of a federated setting. We randomly sample 246,092 sentences for training and 61,719 for testing, where each sentence is described by 10,000 features. We format the task as a 500-class classification problem, and the resulting model has roughly 5 million parameters.
- **MovieLens-100k** [112] is a movie review dataset commonly used for recommendation systems. It contains 100,000 movie ratings from 943 users on 1,682 items ($\approx 6\%$ non-zero entries). We study a (non-convex) matrix factorization task with embedding size 100, thus totaling 262,500 parameters. We treat each non-zero entry as a 'record' for differential privacy, and randomly partition them for training and evaluation.

6.13.2 Hyperparameters

Unless otherwise stated, we fix the following hyperparameters in our experiments: for IMDB, StackOverflow, and MovieLens respectively, we train for 100/50/50 epochs with batch size 64 and privacy $\delta = 10^{-5}/10^{-6}/10^{-6}$. We then perform a grid search on other hyperparameters:

- *Learning rates*: We grid search over {0.03, 0.1, 0.3, 1, 3, 5} for SGD / AdaGrad update rules and from {0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3} for the RMSProp update rule.
- *Per-example clipping thresholds*: We grid search over {0.1, 0.25, 0.5, 1} when performing per-example clipping on clean gradients *without preconditioning* (e.g. for DP-SGD updates), and over {0.1, 0.25, 0.5, 1, 2, 3, 5} when clipping *preconditioned* clean gradients (e.g. for $\texttt{DP}^2$ updates in adaptive iterations). The rationale is that, in general, the preconditioned gradient norms are usually larger than those without preconditioning (recall from Section 6.8.2 that we apply preconditioning *before* privatization in $\texttt{DP}^2$). For AdaDPS and $\texttt{DP}^2$-RMSProp, we also tried a few values of even larger clip thresholds ($\geqslant 10$) though we did not perform a full sweep for other hyperparameters at those values due to computational constraints.
- *Delay parameter s*: For all datasets, s (i.e., the number of optimization steps) is chosen heuristically as a function of the number of steps in an epoch. When reporting the best results (e.g. Figure 6.10, Figure 6.11), we search over $s \in \{195, 390, 780\}$ (roughly 0.5, 1, 2 epochs respectively) for IMDB (390 steps/epoch); $s \in \{100, 300, 1000, 3000\}$ for StackOverflow (3845 steps/epoch); and $s \in \{1250, 15625, 31250, 50000\}$ for MovieLens (1250 steps/epoch).

- *Adaptivity* ϵ: In our settings, the adaptivity parameter ϵ for RMSProp/AdaGrad (in the denominator $D^t = \sqrt{v} + \epsilon$) would affect the amount of adaptivity as well as the norms of preconditioned gradients, which may in turn influence the privacy-utility tradeoff under per-example clipping. We tune ϵ over a small grid of $\{10^{-2}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

All reported results use the best hyperparameter configurations, which are selected using training set metrics (as overfitting generally does not occur under DP noise). To facilitate reproducibility, we summarize the tuned hyperparameters for the main experiments and the ablation studies in Table 6.12 and Table 6.13 below respectively.

Dataset	**DP-SGD**	**DP-RMSProp**	**PDA-DPMD**	**AdaDPS** (w/ RMSProp)	**DP^2-RMSProp**
IMDB	(**5**, 0.5)	(0.3, **0.1**, 10^{-3})	(**5**, 0.5)	(1, **5**, 10^{-3})	(0.1, **3**, 0.5, **5**, **10^{-7}**, **195**)
StackOverflow	(3, 0.25)	(0.03, **0.1**, 10^{-3})	(3, 0.25)	(0.4, **5**, 10^{-3})	(0.3, 0.3, 0.25, **5**, 10^{-5}, 1000)
MovieLens	(0.1, **1**)	(**0.001**, 0.5, 10^{-3})	(0.1, **1**)	(0.01, **10**, **10^{-2}**)	(0.1, 0.03, **1**, **5**, 10^{-3}, 31250)

Table 6.12: **Tuned hyperparameters for different methods across three datasets**. For DP-SGD and PDA-DPMD, the values refer to (LR, clip); for DP-RMSProp and AdaDPS, the values refer to (LR, clip, adaptivity ϵ); and for DP^2, the values refer to (LR for SGD iters, LR for RMSProp iters, clip for SGD iters, clip for RMSProp iters, adaptivity ϵ, delay s). **Bold values** were experimented on the edges of the hyperparameter grids.

Dataset	**Ablation Variant1**	**Ablation Variant 2**
IMDB	(3.0, 0.1, 0.5, 2.0, **10^{-7}**, **780**)	(0.3, 0.3, 0.25, 10^{-3}, **780**)
StackOverflow	(1.0, 1.0, 1.0, 1.0, 10^{-5}, **1000**)	(0.3, **0.001**, 0.25, 10^{-5}, **1000**)

Table 6.13: **Tuned hyperparameters for ablation studies on IMDB and StackOverflow**. Both variants use the RMSProp update rule for the adaptive steps. **Bold values** were experimented on the edges of the hyperparameter grids. For Variant 1 and 2 respectively, the values refer to (LR for SGD iters, LR for RMSProp iters, clip for SGD iters, clip for RMSProp iters, adaptivity ϵ, delay s) and (LR for SGD iters, LR for RMSProp iters, clip for both SGD/RMSProp iters, adaptivity ϵ, delay s). Note that for Variant 2 the clipping threshold do not need to be tuned separately for SGD or RMSProp iters as it applies to preconditioned gradients in both cases.

6.13.3 Results for DP^2-AdaGrad

The DP^2 framework can be applied to a range of adaptive methods beyond RMSProp mostly discussed in the main text. We extend DP^2 to the AdaGrad update rule (with only one line of code change, see Section 6.14), and benchmark its convergence and privacy-utility tradeoffs. In Figure 6.14 and Figure 6.15, the results indicate that DP^2-AdaGrad, like DP^2-RMSProp, can consistently and substantially improve over the baselines in terms of both convergence and absolution performance, demonstrating the generality of DP^2 to other adaptive optimizers.

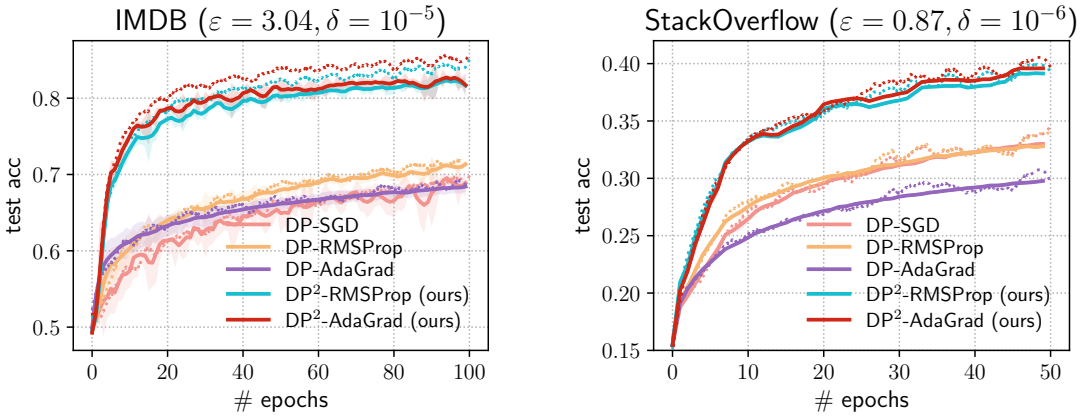


Figure 6.14: (**Extension of Figure 6.10 to the AdaGrad update rule**) Test accuracy of DP^2 compared to DP-SGD, DP-RMSProp, and DP-AdaGrad on IMDB and StackOverflow. Dotted lines denote training performance.

6.13.4 Effects of Increasing Computational Budgets

When differential privacy introduces a large utility gap between private and non-private training, one approach to improving the privacy-utility tradeoff is to increase computational costs by using larger batch sizes under fixed numbers of steps. The noise multiplier needs to increase to achieve the same privacy target, while the overall privacy noise may still be reduced due to the larger batch size. This technique may be adopted in practice when we want to prioritize the utility of private optimization under fixed privacy budgets. In Figure 6.15 (right), we explore the effect of such increased computation on StackOverflow. With a 4$\times$ factor increase in computational cost (4$\times$ larger batch sizes with the same number of training iterations), we observe that the privacy/utility tradeoff of all methods can be substantially improved, narrowing the utility gap to non-private training. In particular, observe that the absolute performance improvement of DP^2 over the vanilla DP baselines remains similar.

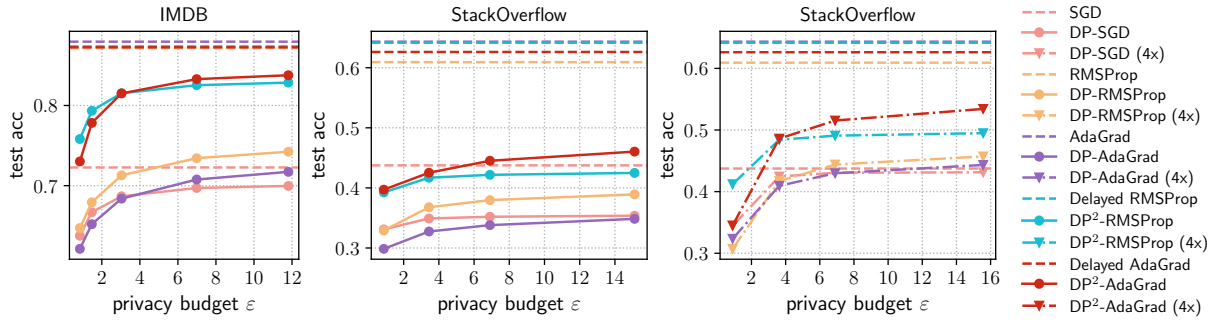


Figure 6.15: (**Extension of Figure 6.11 to the AdaGrad update rule and increased computational cost**) Privacy/utility tradeoffs of DP^2 compared to DP-SGD, DP-RMSProp, and DP-AdaGrad on IMDB and StackOverflow. "(4×)" denotes increasing the batch size and the number of epochs simultaneously by a factor of 4 and picking the appropriate noise multiplier to arrive at similar privacy costs (ε).

6.13.5 Additional Results for Ablation Studies

Table 6.14 summarizes the results for ablation studies on IMDB, StackOverflow, and MovieLens, and Figure 6.16 reports test accuracies on IMDB and StackOverflow during optimization. The variants are discussed in Section 6.10.3 and complete algorithms are presented in Appendix 6.14. We observe that DP^2 indeed consistently outperforms the two (weaker) variants on all datasets, thus verifying our design choices for DP^2. In particular, note that the utility drop of variant 2 (adding noise before preconditioning) on StackOverflow is more significant compared to that on IMDB; we argue that this is due to StackOverflow being a high-dimensional learning task (roughly 5 million model parameters) and thus the detrimental effect of preconditioning per-coordinate noise is larger.

Dataset	**Variant1**	**Variant 2**	**DP^2-RMSProp**
IMDB ↑	.799 ± .006	.643 ± .007	**.815** ± .011
StackOverflow ↑	.382 ± .002	.265 ± .004	**.391** ± .001
MovieLens ↓	3.32 ± .088	3.18 ± .066	**2.78** ± .054

Table 6.14: Summary of ablation studies on all three datasets.

6.13.6 Additional Results for Comparison with Public Data-Assisted Methods

Figure 6.17 extends the results in Section 6.10.2 with convergence plots on IMDB and StackOverflow. On IMDB, we observe that despite not using any auxiliary information, the convergence of DP^2-RMSProp is comparable with that of AdaDPS-RMSProp [187] which uses 1% of training data as the public data (250 examples) to approximate the preconditioner. On StackOverflow where the same public split of 1% corresponds to

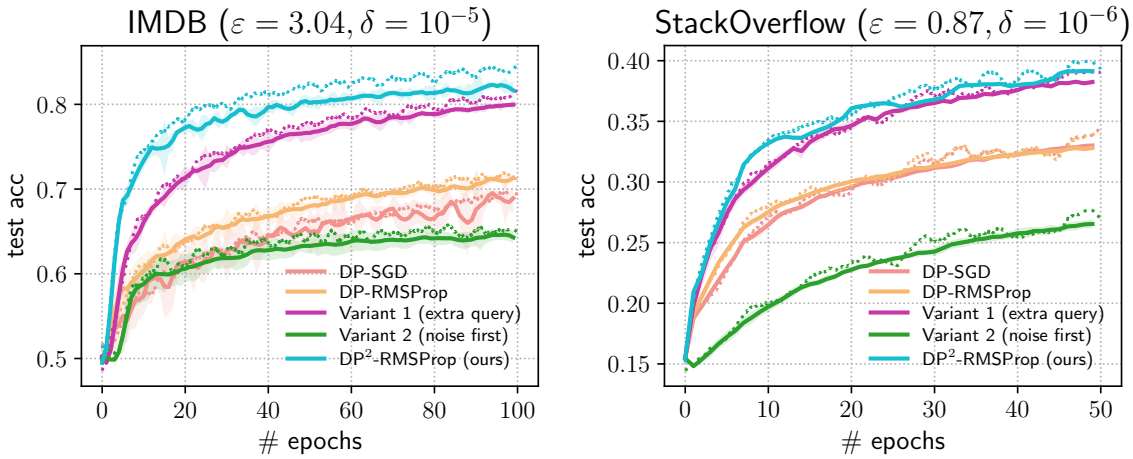


Figure 6.16: Test accuracies for ablation studies on DP^2. Dotted lines correspond to training metrics.

2460 examples, we observe that AdaDPS-RMSProp can outperform DP^2. On the other hand, the extra public data do not help PDA-DPMD outperform DP^2.

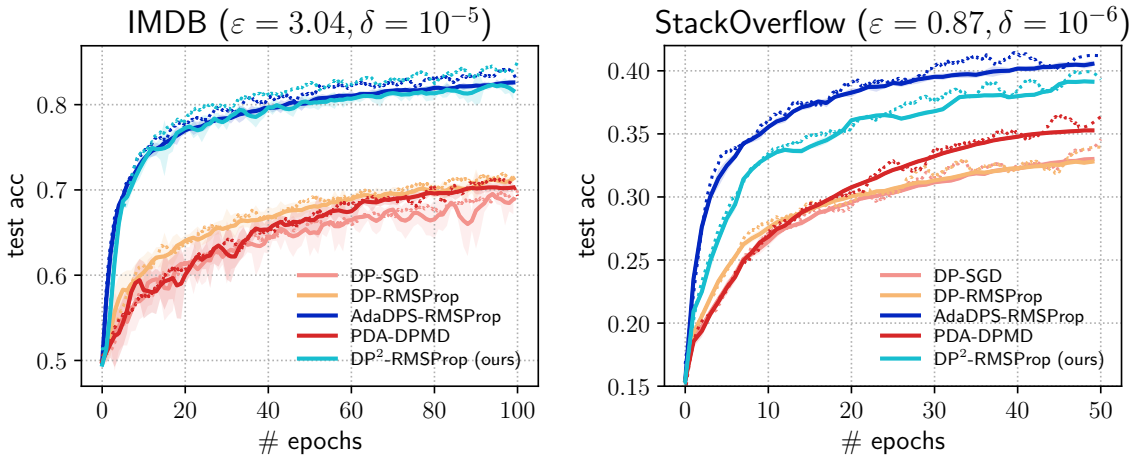


Figure 6.17: Test accuracies of DP^2 compared against recent private (adaptive) methods that leverage public data [15, 187]. Dotted lines correspond to training metrics.

In Figure 6.18, we additionally implement a private AdaGrad method proposed in [142] that also leverages public data. Specifically, in each iteration, the algorithm clips and adds independent noise to both the clean gradients and the preconditioner estimated using clean gradients; it then uses public data to estimate a gradient subspace onto which to project the clipped/noised preconditioner in order to reduce the effect of noise; finally, it preconditions the noisy gradient with the noisy preconditioner and takes an update step. Our implementation differs from [142] in that we use the diagonal form of the preconditioner instead of the full matrix form. To estimate the gradient subspace, we follow the approach described in [333] where the projection matrix $V \in \mathbb{R}^{d \times k}$ where d is the number of parameters and k is the dimension of the subspace is obtained by taking

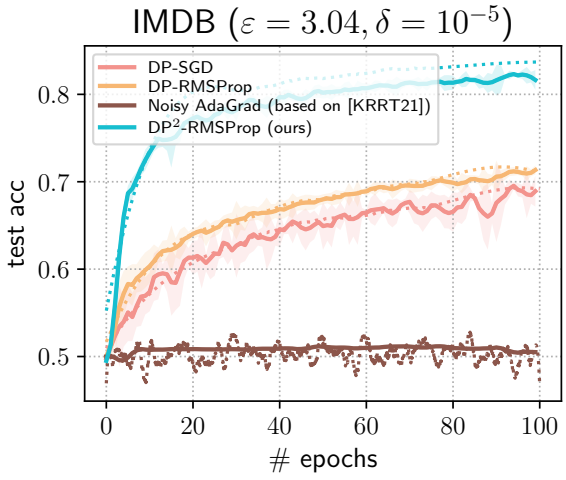


Figure 6.18: Comparing DP^2 against a noisy AdaGrad variant based on [142] where the gradients and the preconditioner are privatized separately.

the top-k eigenspace of M^t with

$$M^t = \frac{1}{|X_{\text{pub}}|} \sum_{x^i \in X_{\text{pub}}} \nabla_{w^t} f\left(x^i; w^t\right) \nabla_{w^t} f\left(x^i; w^t\right)^\top$$

where X_{pub} is the set of public examples. Unfortunately, we have not obtained a satisfactory result for this noisy AdaGrad algorithm. We remark that since the method is extremely computationally expensive (involves computing the eigendecomposition of a $d \times d$ matrix with $d = 10001$ at every iteration), further hyperparameter tuning may help improve the performance. However, our ablation studies (Section 6.10.3 and Appendix 6.13.5) may shed light on the current observations since this method privatizes gradients before preconditioning.

6.14 Algorithms

For completeness, we present all algorithms mentioned in the main text in detail.

- **Non-private version of** DP^2: only changing Line 9 in Algorithm 11 to

$$\tilde{g}^t \leftarrow \frac{1}{b} \sum_{i \in B} \frac{g^{i,t}}{D^t}$$

- DP^2 **with the AdaGrad update rule (**DP^2**-AdaGrad)**: only changing Line 5 in Algorithm 11 to

$$v \leftarrow v + \left(G^t / s_1\right)^2$$

- DP^2 **with Yogi's additive update rule (**DP^2**-Yogi)**: only changing Line 5 in Algorithm 11 to

$$v \leftarrow v + (1-\beta)\text{sign}(G^t/s_1 - v^2)\left(G^t/s_1\right)^2$$

- **Ablation variant 1 (extra query) with delayed preconditioners**: see Algorithm 12. Observe that the clean batch gradients $\{g^{i,t}\}_{i \in B}$ get privatized twice in most iterations (when $(t-1) \bmod s \neq 0$), increasing the total privacy cost.
- **Ablation variant 2 (noise before preconditioning) with delayed preconditioners**: in Line 9 of Figure 11, privatize the batch gradients with the following replacement:

$$\tilde{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(g^{i,t}, C\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C^2\right)\right) / D^t$$

Algorithm 12 Ablation variant 1 (extra query) using delayed preconditioners

Input: T, batch size b, noise multiplier σ, clipping thresholds C_1, C_2, initial model $w^0 \in \mathbb{R}^d$, $v = \mathbf{0}$, constant $\epsilon \in \mathbb{R}_+$, learning rate schedule α^t, moving average parameters β, delay steps s
Set accumulator $G^0 \leftarrow \mathbf{0}$
for $t = 1, \cdots, T$ **do**
  Uniformly randomly sample a mini-batch B with size b from private training data
  Get individual gradients for sample $i \in B$: $g^{i,t} \leftarrow \nabla f(x^i; w^{t-1})$
  Privatize the gradients using the Gaussian mechanism:

$$\tilde{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(g^{i,t}, C_1\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C_1^2\right)\right)$$

  Accumulate the private gradients $\tilde{g}^t$: $G^t \leftarrow G^{t-1} + \tilde{g}^t$
  if $(t-1)$ mod $s = 0$ **then**
    Update moment estimates: $v \leftarrow \beta v + (1-\beta)\left(G^t/s\right)^2$
    Reset accumulator: $G^t \leftarrow \mathbf{0}$
    Set final gradient: $\bar{g}^t \leftarrow \tilde{g}^t$
  else
    Privatize the **clean, preconditioned** gradients using the Gaussian mechanism:

$$\hat{g}^t \leftarrow \frac{1}{b}\left(\sum_{i \in B} \text{clip}\left(\frac{g^{i,t}}{\sqrt{v} + \epsilon}, C_2\right) + \mathcal{N}\left(\mathbf{0}, \sigma^2 C_2^2\right)\right)$$

    Set final gradient: $\bar{g}^t \leftarrow \hat{g}^t$
  end if
  Update model parameters w:

$$w^t \leftarrow w^{t-1} - \alpha^t \bar{g}^t$$

end for
return w^T

Chapter 7

Extensions to General ML Problems: Tilted Empirical Risk Minimization

The q-FFL objective presented in Chapter 4 essentially minimizes the q-norm (q>1) of the loss vector across all clients, which is an approximation of the max norm as q gets larger. Similar intuition leads us to come up with a more general objective based on exponential tilting, which can be used for a variety of ML problems.

7.1 Overview

Many statistical estimation procedures rely on the concept of empirical risk minimization (ERM), in which the parameter of interest, $\theta \in \Theta \subseteq \mathbb{R}^d$, is estimated by minimizing an average loss over the data $\{x_1, \ldots, x_N\}$:

$$\overline{R}(\theta) := \frac{1}{N} \sum_{i \in [N]} f(x_i; \theta) \,. \tag{7.1}$$

Although ERM is widely used in machine learning, it is known to perform poorly in situations where average performance is not an appropriate surrogate for the problem of interest. Significant research has thus been devoted to developing alternatives to traditional ERM for diverse applications, such as learning in the presence of noisy/corrupted data [130, 152], performing classification with imbalanced data [192, 201], ensuring that subgroups within a population are treated fairly [113, 247], or developing solutions with favorable out-of-sample performance [77].

In this chapter, we suggest that deficiencies in ERM can be flexibly addressed via a unified framework, *tilted empirical risk minimization (TERM)*. TERM encompasses a family of objectives, parameterized by a real-valued hyperparameter, t. For $t \in \mathbb{R}^{\setminus 0}$, the t-tilted loss (TERM objective) is given by:

$$\widetilde{R}(t; \theta) := \frac{1}{t} \log \left(\frac{1}{N} \sum_{i \in [N]} e^{t f(x_i; \theta)} \right) . \tag{7.2}$$

TERM generalizes ERM as the 0-tilted loss recovers the average loss, i.e., $\widetilde{R}(0,\theta)=\overline{R}(\theta)$.[1] It also recovers other popular alternatives such as the max-loss ($t\to+\infty$) and min-loss ($t\to-\infty$) (Lemma 25). As we discuss below, although tilted risk minimization is not widely used in machine learning, variants of tilting have been extensively studied in related fields including statistics, applied probability, optimization, and information theory.

7.1.1 Perspectives on Exponential Tilting

We begin by defining *exponential tilting* and discussing uses of tilting in various fields. Let $\mathcal{P} := \{p_\theta\}$ be a set of parametric distributions. For any $x \in \mathcal{X}$, we let $f(x;\theta)$ be the information of x under θ, which is defined as [60]:

$$f(x;\theta) := -\log p_\theta(x). \tag{7.3}$$

Further assume that X is a random variable drawn from distribution $p(\cdot)$, which is not necessarily matched to $\mathcal{P}$, i.e., the model family may be misspecified. The cumulant generating function of the information random variable, $f(X;\theta)$, can be stated as [67, Section 2.2]:

$$\Lambda_X(t;\theta) := \log\left(\mathbb{E}\left[e^{tf(X;\theta)}\right]\right) = \log\sum_x p(x)p_\theta(x)^{-t}, \tag{7.4}$$

where in this chapter $\mathbb{E}[\cdot]$ denotes expectation with respect to the true distribution p unless otherwise stated. This expectation is commonly referred to as an *exponential tilt* of the information density, and can induce parametric distribution shifts that have varied applications in probability, statistics, and information theory. In particular, it is noteworthy that if $\mathcal{P}$ is an exponential family of distributions parameterized by θ, then the tilted distribution $p_\theta(x)^t$ (when normalized by $\int_{\mathcal{X}} p_\theta(x)^t dx$) also belongs to the same exponential family. Further, given samples $\{x_i\}_{i\in[N]}$, the empirical cumulant generating function is defined as:

$$\widetilde{\Lambda}(t;\theta) := \log\left(\frac{1}{N}\sum_{i\in[N]}\left\{e^{tf(x_i;\theta)}\right\}\right). \tag{7.5}$$

It is thus evident that TERM (7.2) can be viewed as an appropriately scaled variant of the empirical cumulant generating function in (7.5). Although tilting of this form has been used in a number of related disciplines, uses of exponential tilting in machine learning are relatively unexplored. We provide several perspectives on exponential tilting from other fields below.

Statistics. Exponential tilting is well-known as a distribution shifting technique in statistics, where the main idea is to draw samples from an exponentially tilted version of the original distribution to improve the convergence properties of statistical estimation,

[1] $\widetilde{R}(0;\theta)$ is defined in (7.20) via the continuous extension of $R(t;\theta)$.

especially when the distribution of interest belongs to an exponential family, such as Gaussian or multinomial. Common use cases include rejection sampling, rare-event simulation, saddle-point approximation [41, p. 156], and importance sampling [260].

Applied Probability. In large deviations theory, exponential tilting lies at the heart of deriving concentration bounds. For example, Chernoff bounds apply Markov's inequality to e^{tX}, which results in a parametric set of bounds by using exponential tilts of various orders. The bound may then be further optimized on the real tilt value to derive the tightest possible bound [67].

Information Theory. While source coding limits and channel capacity are characterized by Shannon entropy and Shannon mutual information (which are simple averages over the information (7.3)) [60], there are other elements of information theory that are not characterized by the average, such as error exponents in channel decoding [95], probability of error in list decoding [212], and computational cost in sequential decoding [17, 204]. These fundamental elements of information theory are asymptotically determined by a non-zero tilted cumulant generating function of the information random variable (7.3) (see [26] for further discussion).

Optimization. Exponential tilting has also appeared as a minimax smoothing approach in optimization [157, 193, 227]. Such smooth approximations to the max often appear through LogSumExp functions, with applications in geometric programming [42, Sec. 9.7], and boosting [203, 255]. We discuss min-max objectives and the connections with TERM in several subsequent sections of the chapter.

Machine Learning. Despite the rich history of tilted objectives in related fields, they have not seen widespread use in ML beyond limited applications such as robust regression [295] and sequential decision making [37, 120]. In this work, we argue that tilting is a critical yet undervalued tool in machine learning. We demonstrate the effectiveness of tilting by (i) rigorously studying properties of the TERM objective, and (ii) exploring its utility for a wide range of ML applications. Surprisingly, we find that this simple extension to ERM can match or exceed state-of-the-art performance from highly tuned, bespoke solutions to common ML problems, from learning with noisy data to ensuring fair performance between subgroups. We highlight several motivating applications of TERM below and provide an outline of the remainder of the chapter in Section 7.1.3.

7.1.2 Motivating Examples

To motivate how the TERM objective (7.2) may be used in machine learning, we provide several running examples below, which are illustrated in Figure 7.1.

(a) Point estimation: As a first example, consider determining a point estimate from a set of samples that contain some outliers. We plot an example 2D dataset in Figure 7.1a, with data centered at (1,1). Using traditional ERM (i.e., TERM with $t = 0$) recovers the

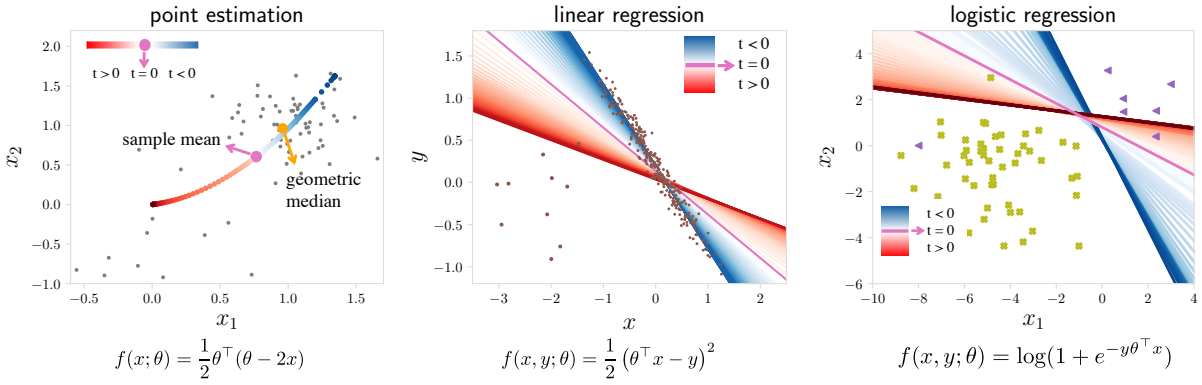


Figure 7.1: Toy examples illustrating TERM as a function of t: (a) finding a point estimate from a set of 2D samples, (b) linear regression with outliers, and (c) logistic regression with imbalanced classes. While positive values of t magnify outliers, negative values suppress them. Setting t=0 recovers the original ERM objective (7.1).

sample mean, which can be biased towards outlier data. By setting $t < 0$, TERM can suppress outliers by reducing the relative impact of the largest losses (i.e., points that are far from the estimate) in (7.2). A specific value of $t < 0$ can in fact approximately recover the geometric median, as the objective in (7.2) can be viewed as approximately optimizing specific loss quantiles (a connection which we make explicit in Section 7.2). In contrast, if these 'outlier' points are important to estimate, setting $t > 0$ will push the solution towards a point that aims to minimize variance, as we prove in Section 7.2, Theorem 25.

(b) Linear regression: A similar interpretation holds for the case of linear regression (Figure 2b). As $t \to -\infty$, TERM finds a line of best while ignoring outliers. However, this solution may not be preferred if we have reason to believe that these 'outliers' should not be ignored. As $t \to +\infty$, TERM recovers the min-max solution, which aims to minimize the worst loss, thus ensuring the model is a reasonable fit for *all* samples (at the expense of possibly being a worse fit for many). Similar criteria have been used, e.g., in defining notions of fairness [113, 247]. We explore several use-cases involving robust regression and fairness in more detail in Section 7.7.

(c) Logistic regression: Finally, we consider a binary classification problem using logistic regression (Figure 2c). For $t \in \mathbb{R}$, the TERM solution varies from the nearest cluster center ($t \to -\infty$), to the logistic regression classifier (t=0), towards a classifier that magnifies the misclassified data ($t \to +\infty$). We note that it is common to modify logistic regression classifiers by adjusting the decision threshold from 0.5, which is equivalent to moving the intercept of the decision boundary. This is fundamentally different than what is offered by TERM (where the slope is changing). As we show in Section 7.7, this added flexibility affords TERM with competitive performance on a number of classification problems, such as those involving noisy data, class imbalance, or a combination of the two.

7.1.3 Contributions

In this work, we explore the use of tilting in machine learning through TERM, a simple, unified framework that can flexibly address various challenges with empirical risk minimization. We first analyze the objective and its solutions, showcasing the behavior of TERM with varying tilt parameters t (Section 7.2). We also establish connections between TERM and related approaches such as distributionally robust optimization in Section 7.3.

We rigorously analyze the relations between TERM and other risks (e.g, Value-at-Risk (VaR), Conditional Value-at-Risk (CVaR), and Entropic Value-at-Risk (EVaR)) in Section 7.4. In particular, we introduce a new risk measure based on TERM, called Tilted Value-at-Risk (TiVaR), to approximate VaR. We show that TiVaR can provide a better approximation of VaR than CVaR in certain regimes, and improves upon EVaR in all regimes.

We develop efficient first-order batch and stochastic methods for solving TERM, both for hierarchical and non-hierarchical cases (Section 7.5 and 7.6). We provide convergence rates scaling with the hyperparameter t on both convex and non-convex problems for both batch and stochastic algorithms. Our solvers run within 2–3$\times$ wall-clock time compared with that of ERM in all explored case studies.

Finally, we show via numerous case studies that TERM is competitive with existing, problem-specific state-of-the-art solutions (Section 7.7). We also extend TERM to handle compound issues, such as the simultaneous existence of noisy samples and imbalanced classes (Section 7.6). Our results demonstrate the effectiveness and versatility of tilted objectives in machine learning.

Outline. This chapter is organized as follows. We discuss general properties and interpretations of TERM in Section 7.2. We connect TERM with other prior risk measures in Section 7.3 and propose a new risk motivated by TERM in Section 7.4. In Section 7.5, we develop both batch and stochastic algorithms for optimizing TERM and provide convergence guarantees for them. We extend TERM to hierarchical multi-objective tilting in Section 7.6 and demonstrate the flexibility and competitive performance of the TERM framework via real-world applications in Section 7.7. We conclude the work in Section 7.8.

7.2 TERM: Properties and Interpretations

To better understand the performance of the t-tilted losses in (7.2), in this section we provide several interpretations of the TERM solutions, leaving the full proofs to the appendix. We make no distributional assumptions on the data, and study properties of TERM under the assumption that the loss function forms a generalized linear model, e.g., L_2 loss and logistic loss. However, we also obtain favorable empirical results using TERM with other objectives such as PCA and deep neural networks in Section 7.7, motivating the extension of this part of our theory beyond GLMs in future work.

7.2.1 Assumptions

We first provide notation and assumptions that are used throughout our theoretical analyses. The results in this chapter are derived under one of the following three nested assumptions (the assumptions become progressively more restrictive, i.e., $3 \rightarrow 2 \rightarrow 1$):

Assumption 8 (Continuous differentiability)**.** *For $i \in [N]$, the loss function $f(x_i; \theta)$ belongs to the differentiability class C^1 (i.e., continuously differentiable) with respect to $\theta \in \Theta \subseteq \mathbb{R}^d$.*

Assumption 9 (Smoothness and strong convexity condition)**.** *Assume that Assumption 8 is satisfied. In addition, for any $i \in [N]$, $f(x_i; \theta)$ belongs to differentiability class C^2 (i.e., twice differentiable with continuous Hessian) with respect to θ. We further assume that there exist $\beta_{\min}, \beta_{\max} \in \mathbb{R}^{>0}$ such that for $i \in [N]$ and any $\theta \in \Theta \subseteq \mathbb{R}^d$,*

$$\beta_{\min} \mathbf{I} \preceq \nabla^2_{\theta\theta^\top} f(x_i; \theta) \preceq \beta_{\max} \mathbf{I}, \tag{7.6}$$

where $\mathbf{I}$ is the identity matrix of appropriate size (in this case $d \times d$), and there does **not** *exist any $\theta \in \Theta$, such that $\nabla_\theta f(x_i; \theta) = 0$ for all $i \in [N]$.*

Assumption 10 (Generalized linear model condition [286])**.** *Assume that Assumption 9 is satisfied. Further, assume that the loss function $f(x; \theta)$ is given by*

$$f(x; \theta) = A(\theta) - \theta^\top T(x), \tag{7.7}$$

where $A(\cdot)$ is a convex function such that there exists $\beta_{\max}$ where for any $\theta \in \Theta \subseteq \mathbb{R}^d$,

$$\beta_{\min} \mathbf{I} \preceq \nabla^2_{\theta\theta^\top} A(\theta) \preceq \beta_{\max} \mathbf{I}\,, \tag{7.8}$$

and

$$\sum_{i \in [N]} T(x_i) T(x_i)^\top \succ 0. \tag{7.9}$$

This set of assumptions become the most restrictive with Assumption 10, which essentially requires that the loss be the negative log-likelihood of an exponential family. While the assumption is stated using the natural parameter of an exponential family for ease of presentation, the results hold for any bijective and smooth reparameterization of the exponential family. For example, Assumption 10 is satisfied by the commonly used L_2 loss for regression and logistic loss for classification (see toy examples (b) and (c) in Figure 7.1). $\sum_{i \in [N]} T(x_i) T(x_i)^\top \succ 0$ assumes a reasonable regularity on the dataset $\{x_i\}_{i \in [N]}$. For instance, in the case of linear regression ($T(x_i) = x_i \in \mathbb{R}^d$), it reduces to the standard regularity assumption $XX^T \succ 0$ (where $X := [x_1, \cdots, x_N] \in \mathbb{R}^{d \times N}$). While Assumption 10 is not satisfied when we use neural network function approximators in Section 7.7, we observe favorable numerical results motivating the extension of these results beyond the cases that are theoretically studied in this work.

In the sequel, many of the results are concerned with characterizing the t-tilted solutions defined as the parametric set of solutions of t-tiled losses by sweeping $t \in \mathbb{R}$,

$$\breve{\theta}(t) \in \arg\min_{\theta \in \Theta} \widetilde{R}(t; \theta), \tag{7.10}$$

where $\Theta \subseteq \mathbb{R}^d$ is an open subset of $\mathbb{R}^d$. Further, let the optimal tilted objective be defined as

$$\widetilde{F}(t) := \widetilde{R}(t; \breve{\theta}(t)). \tag{7.11}$$

We state a final assumption, on $\breve{\theta}(t)$, below.

Assumption 11 (Strict saddle property (Definition 4 in Ge et al. [97]))**.** *We assume that the set* $\arg\min_{\theta\in\Theta} \widetilde{R}(t;\theta)$ *is non-empty for all* $t \in \mathbb{R}$*. Further, we assume that for all* $t \in \mathbb{R}$, $\widetilde{R}(t;\theta)$ *is a "strict saddle" as a function of* θ*, i.e., for all local minima,* $\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta) \succ 0$*, and for all other stationary solutions,* $\lambda_{\min}(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)) < 0$*, where* $\lambda_{\min}(\cdot)$ *is the minimum eigenvalue of the matrix.*

We use the strict saddle property in order to reason about the properties of the t-tilted solutions. In particular, since we are solely interested in the local minima of $\widetilde{R}(t;\theta)$, the strict saddle property implies that for every $\breve{\theta}(t) \in \arg\min_{\theta\in\Theta} \widetilde{R}(t;\theta)$, for a sufficiently small r, for all $\theta \in \mathcal{B}(\breve{\theta}(t), r)$,

$$\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta) \succ 0, \tag{7.12}$$

where $\mathcal{B}(\breve{\theta}(t), r)$ denotes a d-ball of radius r around $\breve{\theta}(t)$. We will show later in Section 7.2.2 that the strict saddle property is readily verified for $t \in \mathbb{R}^{>0}$ under Assumption 9, and we need Assumption 11 to be able to reason about $t \in \mathbb{R}^{<0}$.

7.2.2 General Properties of TERM

We begin by noting several general properties of the TERM objective (7.2). In particular: (i) $\widetilde{R}(t;\theta)$ is L-Lipschitz continuous in θ if $f(x;\theta)$ is L-Lipschitz (Lemma 22); (ii) If $f(x;\theta)$ is strongly convex, the t-tilted loss is strongly convex for $t > 0$ (Lemma 23); and (iii) Given a smooth $f(x;\theta)$, the t-tilted loss is smooth for all finite t (Lemma 24). We state these properties more formally below.

Lemma 22 (Lipschitzness of $\widetilde{R}(t;\theta)$)**.** *For any* $t \in \mathbb{R}$ *and* $\theta \in \Theta$*, if for* $i \in [N]$*,* $f(x_i;\theta)$ *is L-Lipschitz conditnuous in* θ*, then* $\widetilde{R}(t;\theta)$ *is L-Lipschitz in* θ*.*

Lemma 23 (Tilted Hessian and strong convexity for $t \in \mathbb{R}^{>0}$)**.** *Under Assumption 9, for any* $t \in \mathbb{R}$*,*

$$\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta) = \frac{t}{N}\sum_{i\in[N]} (\nabla_\theta f(x_i;\theta) - \nabla_\theta \widetilde{R}(t;\theta))(\nabla_\theta f(x_i;\theta) - \nabla_\theta \widetilde{R}(t;\theta))^\top e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))} \tag{7.13}$$

$$+ \frac{1}{N}\sum_{i\in[N]} \nabla^2_{\theta\theta^\top} f(x_i;\theta) e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}. \tag{7.14}$$

In particular, for all $\theta \in \Theta$ *and all* $t \in \mathbb{R}^{>0}$*, the t-tilted objective is strongly convex. That is*

$$\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta) \succ \beta_{\min}\mathbf{I}. \tag{7.15}$$

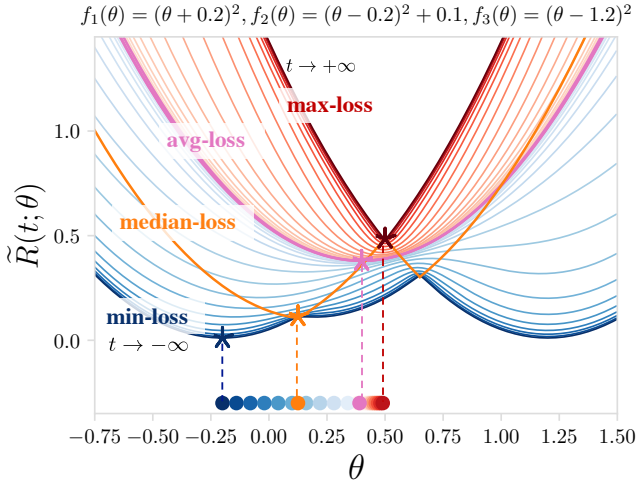


Figure 7.2: TERM objectives for a squared loss with three samples (N=3). As t moves from $-\infty$ to $+\infty$, t-tilted losses recover min-loss, avg-loss, and max-loss. TERM is smooth for all finite t and convex for positive t.

Lemma 22 and 23 are proved in Appendix 7.9. Lemma 23 also implies that under Assumption 9, the strict saddle assumption (Assumption 11) is readily verified.

Lemma 24 (Smoothness of $\widetilde{R}(t;\theta)$)**.** *For any $t \in \mathbb{R}$, let $\beta(t)$ be the smoothness parameter of twice differentiable $\widetilde{R}(t;\theta)$:*

$$\beta(t) := \lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right), \tag{7.16}$$

where $\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)$ is the Hessian of $\widetilde{R}(t;\theta)$ at θ and $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue. Under Assumption 9, for any $t \in \mathbb{R}$, $\widetilde{R}(t;\theta)$ is a $\beta(t)$-smooth function of θ. Further, for $t \in \mathbb{R}^{\leqslant 0}$,[2]

$$\beta(t) < \beta_{\max}, \tag{7.17}$$

where $\beta_{\max}$ is defined in Assumption 9. For $t \in \mathbb{R}^{>0}$,

$$0 < \lim_{t\to+\infty} \frac{\beta(t)}{t} < +\infty. \tag{7.18}$$

Lemma 24 (proved in Appendix 7.9.1) indicates that t-tilted losses are $\beta(t)$-smooth for all t. $\beta(t)$ is bounded for all negative t and moderately positive t, whereas it scales linearly with t as $t \to +\infty$, which has been previously studied in the context of exponential smoothing of the max [157, 227]. This can also be observed visually via the toy example in Figure 7.2.

As discussed in Section 7.1, TERM can recover traditional ERM (t=0), the max-loss ($t\to+\infty$), and the min-loss ($t\to-\infty$). We formally state this in Lemma 25 below.

Lemma 25. *Under Assumption 8,*

$$\widetilde{R}(-\infty;\theta) := \lim_{t\to-\infty} \widetilde{R}(t;\theta) = \breve{R}(\theta), \tag{7.19}$$

[2] $\mathbb{R}^{\leqslant 0}$ denotes the set of non-positive real numbers.

$$\widetilde{R}(0;\theta) := \lim_{t\to 0} \widetilde{R}(t;\theta) = \overline{R}(\theta), \tag{7.20}$$

$$\widetilde{R}(+\infty;\theta) := \lim_{t\to +\infty} \widetilde{R}(t;\theta) = \widehat{R}(\theta), \tag{7.21}$$

where $\widehat{R}(\theta)$ is the max-loss and $\breve{R}(\theta)$ is the min-loss[3]*:*

$$\widehat{R}(\theta) := \max_{i\in[N]} f(x_i;\theta), \qquad \breve{R}(\theta) := \min_{i\in[N]} f(x_i;\theta). \tag{7.22}$$

Note that Lemma 25 has been studied or observed before in the entropic risk literature [e.g., 9], as well as other contexts [55]. This lemma also implies that $\breve{\theta}(0)$ is the ERM solution, $\breve{\theta}(+\infty)$ is the min-max solution, and $\breve{\theta}(-\infty)$ is the min-min solution. In other words, a benefit of TERM is that it offers a continuum of solutions between the min and max losses.

Providing a smooth tradeoff between these specific losses can be beneficial for a number of practical use-cases—both in terms of the resulting solution and the difficulty of solving the problem itself. We empirically demonstrate the benefits of such a tradeoff in Section 7.7. We also visualize the solutions to TERM for a toy problem in Figure 7.2, which allows us to illustrate several special cases of the general framework. Interestingly, we additionally show that the TERM solution can be viewed as a smooth approximation to the *tail probability of losses*, which effectively minimizes quantiles of losses such as the median loss (Section 7.4). In Figure 7.2, it is clear to see why this may be beneficial, as the median loss (orange) can be highly non-smooth in practice. In Theorem 23 and 24 below, we formally characterize how tilted objectives change as a function of values t (proofs provided in Appendix 7.9).

Theorem 23 (Tilted objective is increasing with t)**.** *Under Assumption 10, for all $t \in \mathbb{R}$, and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\widetilde{R}(t;\theta) \geqslant 0. \tag{7.23}$$

Theorem 24 (Optimal tilted objective is increasing with t)**.** *Under Assumption 10, for all $t \in \mathbb{R}$, and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\widetilde{F}(t) = \frac{\partial}{\partial t}\widetilde{R}(t;\breve{\theta}(t)) \geqslant 0. \tag{7.24}$$

Recall that TERM as $t \to -\infty$ and $t \to \infty$ corresponds to min-loss and max-loss, respectively. We discuss in Section 7.4.2 that solving TERM with any $t \in \mathbb{R}$ can indeed be viewed as approximately minimizing the k-th smallest loss ($k \in [N]$) among all N individual losses. As we increase k from 1 to N, the corresponding value of t sweeps in $(-\infty, \infty)$. Theorem 24 hence roughly states that the optimal k-th smallest loss is non-decreasing with k, which is intuitively expected.

We next provide two interesting interpretations of the TERM framework to further understand its behavior.

[3]When the argument of the max-loss or the min-loss is not unique, for the purpose of differentiating the loss function, we define $\widehat{R}(\theta)$ as the average of the individual losses that achieve the maximum, and $\breve{R}(\theta)$ as the average of the individual losses that achieve the minimum.

7.2.3 Interpretation 1: Re-Weighting Samples to Magnify/Suppress Outliers

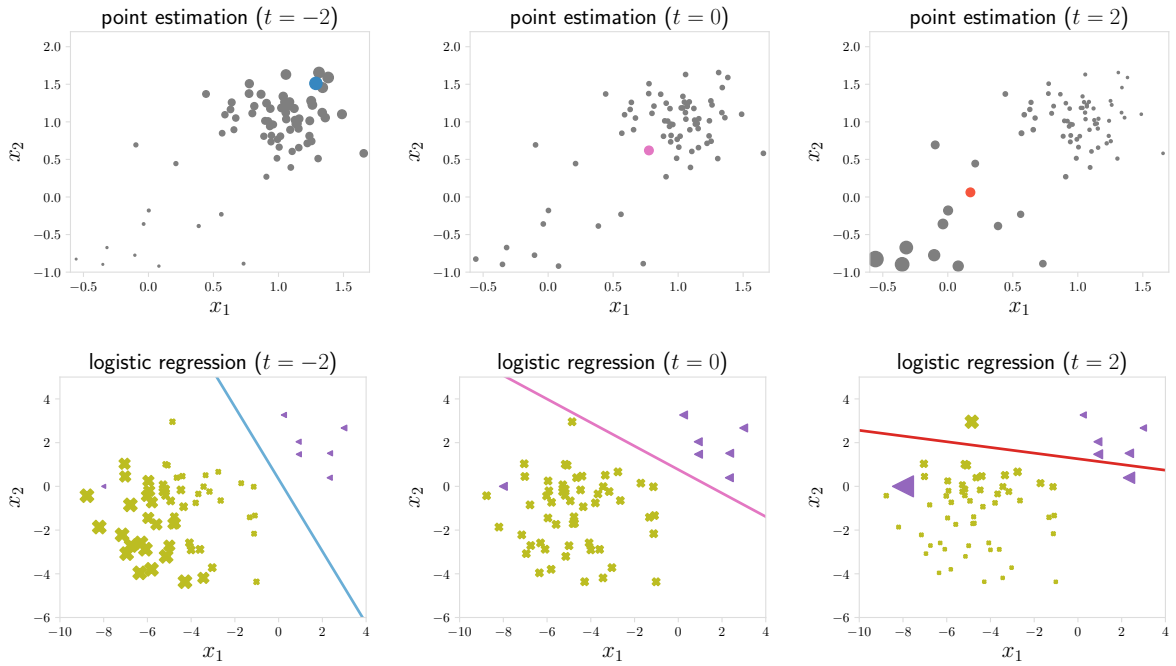


Figure 7.3: We visualize the size of the samples using their gradient weights. Negative t's ($t = -2$ on the left) focus on the inlier samples (suppressing outliers), while positive t's ($t = 2$ on the right) magnify the outlier samples.

As discussed via the toy examples in Section 1, TERM can be tuned (using t) to magnify or suppress the influence of outliers. We make this notion rigorous by exploring the *gradient* of the t-tilted loss in order to reason about the solutions to the objective defined in (7.2).

Lemma 26 (Tilted gradient). *For a smooth loss function $f(x;\theta)$,*

$$\nabla_\theta \widetilde{R}(t;\theta) = \sum_{i\in[N]} w_i(t;\theta)\nabla_\theta f(x_i;\theta), \tag{7.25}$$

where tilted weights are given by

$$w_i(t;\theta) := \frac{e^{tf(x_i;\theta)}}{\sum_{j\in[N]} e^{tf(x_j;\theta)}} = \frac{1}{N} e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}. \tag{7.26}$$

Proof. Under Assumption 8, we have:

$$\nabla_\theta \widetilde{R}(t;\theta) = \nabla_\theta \left\{ \frac{1}{t} \log \left(\frac{1}{N} \sum_{i\in[N]} e^{tf(x_i;\theta)} \right) \right\} = \frac{\sum_{i\in[N]} \nabla_\theta f(x_i;\theta) e^{tf(x_i;\theta)}}{\sum_{i\in[N]} e^{tf(x_i;\theta)}}. \tag{7.27}$$

□

Lemma 26 provides the gradient of the tilted objective, which has been studied previously in the context of exponential smoothing (see Pee and Royset [227, Proposition 2.1]). From this, we can observe that the tilted gradient is a weighted average of the gradients of the original individual losses, where each data point is weighted exponentially proportional to the value of its loss. Note that $t = 0$ recovers the uniform weighting associated with ERM, i.e., $w_i(t;\theta) = 1/N$. For positive t, this has the effect of *magnifying* the outliers—samples with large losses—by assigning more weight to them, and for negative t, it *suppresses* the outliers by assigning less weight to them (Figure 7.3).

Generalizing the notion of tilted gradients (weighted average of individual gradients), we define tilted empirical mean over any N-vector $\mathbf{u} \in \mathbf{R}^N$ below, which will be used throughout the chapter.

Definition 13 (Tilted empirical mean and variance)**.** *For $\mathbf{u} \in \mathbf{R}^N$, let weighted empirical mean with weights $\mathbf{w} \in \Delta^N$ (where Δ^N stands for N dimensional simplex) be defined as*

$$\widehat{E}_{\mathbf{w}}(\mathbf{u}) := \sum_{i\in[N]} w_i u_i. \tag{7.28}$$

Tilted empirical mean is weighted empirical mean with tilted weights, i.e.,

$$\widehat{E}_{\mathbf{w}(t;\theta)}(\mathbf{u}) := \sum_{i\in[N]} w_i(t;\theta) u_i, \tag{7.29}$$

$$\widehat{E}_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{u}) := \sum_{i\in[N]} w_i(t;\breve{\theta}(t)) u_i, \;\; \widehat{E}_t := \widehat{E}_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{u}), \tag{7.30}$$

where $w_i(t;\theta)$ is defined in Eq. (7.26)*, and $\breve{\theta}(t)$ is defined in Eq.* (7.10)*. We also refer to $\widehat{E}_t$ as the "t-tilted empirical mean". Similarly, tilted empirical variance is defined as*

$$\widehat{\mathbf{Var}}_{\mathbf{w}(t;\theta)}(\mathbf{u}) := \widehat{E}_{\mathbf{w}(t;\theta(t))}\left(u_i - \widehat{E}_{\mathbf{w}(t;\theta(t))}(\mathbf{u})\right)^2, \tag{7.31}$$

$$\widehat{\mathbf{Var}}_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{u}) := \widehat{E}_t(u_i - \widehat{E}_t(\mathbf{u}))^2, \;\; \widehat{\mathbf{Var}}_t := \widehat{\mathbf{Var}}_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{u}), \tag{7.32}$$

and we refer to $\widehat{\mathbf{Var}}_t$ as the "t-tilted empirical variance".

As discussed before, the full gradient of TERM is tilted empirical mean of individual gradients $\{\nabla_\theta f(x_i;\theta)\}_{i\in[N]}$ with weights proportional to $e^{tf(x_i;\theta)}$. In the next section as well as Appendix 7.9.3, we will prove other properties of TERM using tilted empirical mean and variance defined here.

7.2.4 Interpretation 2: Empirical Bias/Variance tradeoff

Another key property of the TERM solutions is that for any $t \in \mathbb{R}$, t-tilted empirical variance of the losses across all samples will decrease if we increase t by a small amount of value. We formally stated this in Theorem 25.

Theorem 25 (Variance reduction)**.** *Let* $\mathbf{f}(\theta) := (f(x_1;\theta)),\ldots,f(x_N;\theta))$*. Then, under Assumption 10 and Assumption 11, for any* $t \in \mathbb{R}$*,*

$$\frac{\partial}{\partial t}\left\{\widehat{\mathbf{Var}}_\tau(\mathbf{f}(\breve{\theta}(t)))\right\}\Big|_{t=\tau} < 0. \tag{7.33}$$

Note that $\widehat{\mathbf{Var}}_\tau$ is τ-tilted empirical variance defined in Eq. (7.32). Hence, for any t, the t-tilted empirical variance among N losses will decrease if we increase t by a small value. When $\tau = 0$, $\widehat{\mathbf{Var}}_\tau$ reduces to standard empirical variance. In particular, Theorem 25 states that the empirical variance of the loss vector decreases if t is chosen to be a small positive value. Therefore, it is possible to trade off between optimizing the average loss vs. reducing variance, allowing the solutions to potentially achieve a better bias-variance tradeoff for generalization [27, 118, 205]. At a high level, this property is consistent with and extends the approximation of TERM mentioned by Liu and Theodorou [193, Section V.A], which approximates TERM as the empirical risk regularized with variance of the loss at $t = 0$. We rely on this property to achieve better generalization in classification in Section 7.7.

In addition to empirical variance across all losses, there are other related distribution uniformity measures. In Theorem 26 below, we also prove that entropy of the weight distribution at solution $\breve{\theta}(t)$ tilted by τ close to t is increasing with t, which indicates that larger t's encourages more uniform solutions measured via entropy.

Theorem 26 (Gradient weights become more uniform by increasing t)**.** *Under Assumption 10 and Assumption 11, for any* $t \in \mathbb{R}^{>0}$*,*

$$\frac{\partial}{\partial t}H(\mathbf{w}(\tau;\breve{\theta}(t)))\Big|_{\tau=t} > 0, \tag{7.34}$$

where $H(\cdot)$ *denotes the Shannon entropy function measured in nats,*

$$H\left(\mathbf{w}(t;\theta)\right) := -\sum_{i\in[N]} w_i(t;\theta)\log w_i(t;\theta). \tag{7.35}$$

Full proofs of the theorems presented in this section can be found in Appendix 7.9.3. In the next section, we connect TERM to other objectives. Note that the results in all subsequent sections do not require the GLMs assumption, unless stated otherwise.

7.3 Connections to Other Risk Measures

In this section (and subsequently in Section 7.4) we explore TERM by comparing, contrasting, and drawing connections between TERM and other common risk measures. To do so, we first introduce a distributional version of TERM, which is closely related to entropic risk (measure) in previous literature [9, 94]. Entropic risk, denoted as $R_X(t;\theta)$, can be viewed as the scaled cumulant generating function of $f(X;\theta)$, i.e.,

$$R_X(t;\theta) := \frac{1}{t}\Lambda_X(t;\theta) = \frac{1}{t}\log\left(\mathbb{E}\left[e^{tf(X;\theta)}\right]\right) = \frac{1}{t}\log\sum_x p(x)p_\theta(x)^{-t}. \tag{7.36}$$

We note that entropic risk is usually defined over $t \in \mathbb{R}^{>0}$ in the literature [94]. In Eq. (7.36) above, we naturally extend its definition to support $t \in \mathbb{R}$. The TERM objective $\widetilde{R}(t;\theta)$ is the empirical version of entropic risk $R_X(t;\theta)$ ($t \in \mathbb{R}$). One of the contributions of this work can be viewed as providing an operational meaning to the value of the (empirical) entropic risk and rigorously investigating its properties for $t \in \mathbb{R}^{<0}$. In the next sections (Section 7.3.1–Section 7.3.3), we characterize various relations between tilted risks (TERM or entropic risk) and other common risk measures, both in terms of the empirical variants (involving TERM) and distributional forms (involving entropic risk).

7.3.1 TERM and Rényi Cross Entropy

We begin by demonstrating that TERM can be viewed as form of Rényi cross entropy minimization, which helps to explain the uniformity properties of TERM discussed in Section 7.2.4. Consider the cross entropy between p and p_θ defined by

$$H(p\|p_\theta) := \mathbb{E}\left[f(X;\theta)\right] = \sum_x p(x)\log\left(\frac{1}{p_\theta(x)}\right). \tag{7.37}$$

Hence, minimizing $\mathbb{E}\left[f(X;\theta)\right]$ is equivalent to minimizing the cross entropy between the true distribution and the postulated distribution. The empirical variant of (7.37) would be empirical risk minimization (7.1).

For $\rho \in \mathbb{R}^{>0}$, let Rényi cross entropy of order ρ between p and q be defined as:[4]

$$H_\rho(p\|q) := \frac{1}{1-\rho}\log\left(\sum_x p(x)q(x)^{\rho-1}\right). \tag{7.38}$$

Rényi cross entropy can be viewed as a natural extension of cross entropy, and in fact it recovers cross entropy for $\rho = 1$, i.e., $H_1(p\|q) = H(p\|q)$. Rényi cross-entropy can also be viewed as a natural extension of Rényi entropy, which it recovers when $p = q$, i.e., $H_\rho(p\|p) = H_\rho(p)$, where Rényi entropy of order ρ is defined as

$$H_\rho(p) := \frac{1}{1-\rho}\log\left(\sum_x p(x)^\rho\right). \tag{7.39}$$

It is straightforward to see that the entropic risk can be expressed in terms of Rényi cross entropy:

$$R_X(t;\theta) = H_{1-t}(p\|p_\theta). \tag{7.40}$$

[4] H_1 is defined via continuous extension.

Equivalently, in the empirical world, TERM can be expressed as:

$$\widetilde{R}(t;\theta) = H_{1-t}(\mathbf{u}\|\mathbf{w}(1;\theta)), \tag{7.41}$$

where $\mathbf{u}$ denotes the uniform N-vector and $\mathbf{w}(1;\theta) := (w_1(1;\theta),\ldots,w_n(1;\theta))$ with $w_i(1;\theta)$ defined in Eq. (7.26), and for any two N-vectors $\mathbf{p}$ and $\mathbf{q}$,

$$H_\rho(\mathbf{p}\|\mathbf{q}) := \frac{1}{1-\rho}\log\left(\sum_{i\in N} p_i q_i^{\rho-1}\right). \tag{7.42}$$

In other words, if we treat the loss $f(x_i;\theta)$ as log-likelihood of the sample x_i under p_θ, this implies that TERM is the Rényi entropy of order $(1-t)$ between the uniform vector and the normalized likelihood vector of all samples, $\mathbf{w}(1;\theta)$. Hence, minimizing over θ is encouraging the *uniformity* of $\mathbf{w}(1;\theta)$ in the sense of the Rényi cross entropy with the uniform vector.

7.3.2 TERM as a Regularizer to Empirical Risk

TERM can also be interpreted as a form of regularization in traditional ERM. We first note that by Taylor series expansion at $t = 0$, TERM can be approximately decomposed into empirical risk regularized by t times the empirical variance of the loss, for small t [193, Section V.A]. Here, we provide an exact interpretation of TERM as regularized ERM for all t. We first look at the distributional case, i.e., relating $R_X(t;\theta)$ to cross entropy as follows.

Lemma 27. *The entropic risk of order t can be stated as:*

$$R_X(t;\theta) = H(p\|p_\theta) + \frac{1}{t}D(p\|T(p,p_\theta,-t)), \tag{7.43}$$

where D denotes KL divergence between two distributions and $T(p,p_\theta,-t)$ is a mismatched tilted distribution defined as [246, Definition 1]

$$T(p,p_\theta,-t)(x) := \frac{p(x)p_\theta(x)^{-t}}{\sum_u p(u)p_\theta(u)^{-t}}. \tag{7.44}$$

Proof. Consider the following equation:

$$\sum_x p(x)\log\left(\frac{p(x)}{T(p,p_\theta,-t)(x)}\right) = -t\sum_x p(x)\log\frac{1}{p_\theta(x)} + \log\left(\sum_x p(x)p_\theta(x)^{-t}\right), \tag{7.45}$$

which directly implies the desired identity. □

In other words, entropic risk of order t is equivalent to the cross entropy risk regularized via a tilted mismatched distribution. Let $\mathbf{w}(t;\theta) := (w_1(t;\theta),\ldots,w_n(t;\theta))$ denote the tilted weight vector of the n samples. Our next result is an empirical variant of Lemma 27.

Lemma 28. *TERM objective can be restated as follows:*

$$\widetilde{R}(t;\theta) = \bar{R}(\theta) + \frac{1}{t} D(\mathbf{u} \| \mathbf{w}(t;\theta)), \tag{7.46}$$

where $\bar{R}(\theta)$ *is the empirical risk* (7.1), $\mathbf{u}$ *denotes the uniform* N*-vector, i.e.,* $\mathbf{u} := \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$, *and where for* N*-vectors* $\mathbf{p}$ *and* $\mathbf{q}$,

$$D(\mathbf{p} \| \mathbf{q}) := \sum_{i \in [N]} p_i \log\left(\frac{p_i}{q_i}\right). \tag{7.47}$$

Proof. The proof is a consequence of the following identity:

$$\frac{1}{t}\frac{1}{N}\sum_{i\in[N]} \log\left(\frac{\frac{1}{N}}{w_i(t;\theta)}\right) + \frac{1}{N}\sum_{i\in[N]} f(x_i;\theta) = \frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}\right). \tag{7.48}$$

□

Hence, TERM aims to minimize an average loss regularized by the KL divergence between the weight vector (which exponentially tilts the individual losses) and the uniform vector.

7.3.3 TERM and Distributionally Robust Risks

Finally, we note that TERM is closely related to distributionally robust optimization (DRO) objectives [e.g., 51, 76, 77, 105, 219]. In particular, TERM with $t > 0$ is equivalent to a form of DRO with a max-entropy regularizer, i.e., the constraint set is determined by a KL ball around uniform distribution [93, 233, 254]:

$$\widetilde{R}(t;\theta) = \max_{q \in \Delta_N} \left\{ \sum q_i f(x_i;\theta) - \frac{1}{t}\sum_{i\in[N]} q_i \log N q_i \right\} = \max_{q\in\Delta_N} \left\{ H(\mathbf{q}\|\mathbf{w}(1;\theta)) - \frac{1}{t} D(\mathbf{q}\|\mathbf{u}) \right\}, \tag{7.49}$$

and the corresponding relations in the distributional form is

$$R_X(t;\theta) = \max_q \left\{ \mathbb{E}_q[f(X;\theta)] - \frac{1}{t} D(q\|p) \right\} = \max_q \left\{ H(q\|p_\theta) - \frac{1}{t} D(q\|p) \right\}. \tag{7.50}$$

This relation is also a special case of Donsker-Varadhan Variational Formula [81]. We note that similar connections between DRO and TERM have also been explored in concurrent works by Qi et al. [232, 233] specifically in the limited context of stochastic optimization methods for solving class imbalance with $t > 0$.

In the next section, we propose a new risk motivated by TERM, which may be of independent interest.

7.4 Tilted Value-at-Risk and Value-at-Risk

In this section we provide connections between TERM and risk measures such as Value-at-Risk (VAR) that specifically target loss quantiles. In particular, based on TERM, we propose a new risk—*Tilted Value-at-Risk (TiVaR)* and discuss its relations with existing risks (Section 7.4.2). We find that TiVaR is a computationally efficient alternative to VaR that provides tighter approximations to VaR than prior risks, which again helps to motivate the use of TERM.

7.4.1 Tail Probabilities of Losses and Value-at-Risk (VaR)

The tail probabilities of losses focus on quantiles of losses that exceed a certain threshold, as formally defined below.

Definition 14 (Tail probability of losses)**.** *For all $\gamma \in \mathbb{R}$, let $Q_X(\gamma;\theta)$ denote the probability of the losses $f(X;\theta)$ no smaller than γ, i.e.,*

$$Q_X(\gamma;\theta) := P\left[f(X;\theta) \geqslant \gamma\right]. \tag{7.51}$$

Equivalently, define the empirical variant $\widetilde{Q}(\gamma;\theta)$ over samples x_i for $i \in [N]$:

$$\widetilde{Q}(\gamma;\theta) := \frac{1}{N}\sum_{i\in[N]} \mathbb{I}\left\{f(x_i;\theta) \geqslant \gamma\right\} \tag{7.52}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function.

Notice that $\widetilde{Q}(\gamma;\theta) \in \left\{0, \frac{1}{N}, \ldots, 1\right\}$ quantifies the fraction of the data for which loss is at least γ. For example, optimizing for 90% of the individual losses (ignoring the worst-performing 10%) could be a more reasonable practical objective than the pessimistic min-max objective. Another common application of this is to use the median in contrast to the mean in the presence of noisy outliers.

Using tail distribution of losses, Value-at-Risk (VaR) [138] with confidence α ($0 < \alpha < 1$) is defined as

$$\mathrm{VaR}_X(1-\alpha;\theta) := \min_{\gamma}\left\{\gamma : Q_X(\gamma;\theta) \leqslant \alpha\right\}, \tag{7.53}$$

and the empirical variant for $\alpha \in \{\frac{k}{N}\}_{k\in[N]}$ is

$$\widetilde{\mathrm{VaR}}(1-\alpha;\theta) := \min_{\gamma}\left\{\gamma : \widetilde{Q}(\gamma;\theta) \leqslant \alpha\right\}. \tag{7.54}$$

Notice that when we view the loss as log-likelihood of a parametric probability distribution function, $Q_X(\gamma;\theta)$ (Definition 14) can be viewed as the complementary cumulative distribution function (CDF) of the information random variable $f(X;\theta)$. Given

the definition of VaR, $Q_X(\gamma;\theta)$ can also be viewed as 'inverted' VaR, as we formalize and prove in Lemma 29 and 30 below. Let

$$Q_X^0(\gamma) := \min_\theta Q_X(\gamma;\theta), \quad \theta_X^0(\gamma) \in \arg\min_\theta Q_X(\gamma;\theta), \tag{7.55}$$

$$\widetilde{Q}^0(\gamma) := \min_\theta \widetilde{Q}(\gamma;\theta), \quad \theta^0(\gamma) \in \arg\min_\theta \widetilde{Q}(\gamma;\theta). \tag{7.56}$$

where Q_X and $\widetilde{Q}$ is defined in Definition 14. Optimizing $\widetilde{Q}(\gamma;\theta)$ is equivalent to optimizing VaR. Formally, we have the following lemmas.

Lemma 29. *Assume* $\min_\theta Q_X(\gamma;\theta)$ *is strictly decreasing with* γ*. We note*

$$\min_\theta \left\{ VaR_X(1 - Q_X^0(\gamma);\theta) \right\} = \gamma, \quad \arg\min_\theta \left\{ VaR_X(1 - Q_X^0(\gamma);\theta) \right\} \ni \theta_X^0(\gamma). \tag{7.57}$$

Note that $\min_\theta Q_X(\gamma;\theta)$ is non-increasing as γ increases by definition. The additional strict monotonic assumption on $\gamma \mapsto \min_\theta Q_X(\gamma;\theta)$ can be easily satisfied if $f(X;\theta)$ is a continuous random variable and γ is in the range of f. Lemma 29 is proved as follows.

Proof. First, we note for any θ and γ_0 such that $Q_X(\gamma_0;\theta) \leqslant Q_X^0(\gamma)$, we have $\gamma_0 \geqslant \gamma$. Otherwise, there exist $\theta', \gamma' < \gamma$ and $Q_X(\gamma';\theta') \leqslant Q_X^0(\gamma)$, which in turn implies that

$$\min_\theta P[f(X;\theta) \geqslant \gamma'] \leqslant P[f(X;\theta') \geqslant \gamma'] \leqslant Q_X^0(\gamma), \tag{7.58}$$

contradicting $Q_X^0(\gamma') > Q_X^0(\gamma)$. The proof completes combining with the fact that the function value of $\mathrm{VaR}_X(1 - Q_X^0(\gamma);\theta)$ can achieve γ at any $\theta_X^0(\gamma)$. □

Lemma 30 below describes the empirical variant, which does not require the strict monotonic assumption.

Lemma 30. *For any* $\gamma \in (\widetilde{F}(-\infty), \widetilde{F}(+\infty))$ *where* $\widetilde{F}(t)$ *is defined as the optimal tilted objective as in Eq.* (7.11)*, let* $\gamma^0 = \min\left\{\gamma' | \widetilde{Q}^0(\gamma') = \widetilde{Q}^0(\gamma)\right\}$*. Then*

$$\min_\theta \left\{ \widetilde{VaR}(1 - \widetilde{Q}^0(\gamma^0);\theta) \right\} = \gamma^0, \quad \arg\min_\theta \left\{ \widetilde{VaR}(1 - \widetilde{Q}^0(\gamma^0);\theta) \right\} \ni \theta^0(\gamma^0). \tag{7.59}$$

Both tail distribution of losses and VaR are usually non-smooth and non-convex, and solving them to global optimality is very challenging. In the next section, we show that TiVaR (an objective based on TERM) provides a good upper bound on VaR, and is computationally more efficient, as VaR is not even continuous. In parallel, in Appendix 7.10, we prove that TERM also provides a reasonable approximate solution to the minimizer of tail probability of losses (i.e., inverted VaR).

The proof of one of the main theorems of this section (Theorem 36) relies on a new variant of Chernoff bound for non-negative random variables, which may be of independent interest.

Theorem 27 (Chernoff bound for non-negative random variables)**.** *Let X be a non-negative random variable. Further assume that $E\left[e^{tX}\right] < \infty$ for all $t \in \mathbb{R}$. Then for $\gamma > 0$,*

$$P[X \geqslant \gamma] \leqslant \inf_{t \in \mathbb{R}} \left\{ \frac{E\left[e^{tX}\right] - 1}{e^{t\gamma} - 1} \right\} \leqslant \inf_{t \in \mathbb{R}^+} \left\{ \frac{E\left[e^{tX}\right]}{e^{t\gamma}} \right\}, \tag{7.60}$$

where the latter term is the generic Chernoff bound with $\gamma > 0$.

Proof. The theorem holds by applying Markov's inequality twice on $e^{tX} - 1$ $(t \geqslant 0)$ and $1 - e^{tX}$ $(t < 0)$, and noting that

$$P[X \geqslant \gamma] \leqslant \min \left\{ \inf_{t \in \mathbb{R}^{\geqslant 0}} \left\{ \frac{E\left[e^{tX}\right] - 1}{e^{t\gamma} - 1} \right\}, \inf_{t \in \mathbb{R}^-} \left\{ \frac{1 - E\left[e^{tX}\right]}{1 - e^{t\gamma}} \right\} \right\} = \inf_{t \in \mathbb{R}} \left\{ \frac{E\left[e^{tX}\right] - 1}{e^{t\gamma} - 1} \right\}.$$

□

Theorem 27 presents a tighter Chernoff bound for non-negative random variables. To the best of our knowledge, despite the fact that this bound is a simple extension of the generic Chernoff bound, and the existing variants of Chernoff bounds in prior works [38, 305], we have not seen the result we have here appear elsewhere in this form. In particular, notice that the search for an optimal value of t has been extended from non-negative values to all real numbers. This can result in significantly tighter bounds, especially in small deviations regime, as visualized empirically on two simple distributions in Figure 7.15, Appendix 7.10. We will see how this leads to significantly better bounds in robustness applications.

7.4.2 TiVaR: Tilted Value-at-Risk

In this section, we introduce a new risk measure, called Tilted Value-at-Risk (TiVaR). To put TiVaR in perspective, we briefly state other existing risks first. Conditional Value-at-Risk (CVaR) minimizes the average risk of tail events where the risk is above some threshold [242, 243]. One form of CVaR is

$$\text{CVaR}_X(1 - \alpha; \theta) := \min_{\gamma} \left\{ \gamma + \frac{1}{\alpha} \mathbb{E}[f(X; \theta) - \gamma]_+ \right\}. \tag{7.61}$$

It is worth noting that $\text{CVaR}_X(1 - \alpha; \theta)$ is a dual formulation of DRO with an uncertainty set that perturbs arbitrary parts of the data by an amount up to $\frac{1}{\alpha}$ [61, 243]. Formally, the dual of DRO $\max_{Q: \left\{ \frac{dQ}{dP} \leqslant \frac{1}{\alpha} \right\}} \mathbb{E}_Q[f(X; \theta)]$ is $\text{CVaR}_X(1 - \alpha; \theta) = \min_\gamma \left\{ \gamma + \frac{1}{\alpha} \mathbb{E}[f(X; \theta) - \gamma]_+ \right\}$. Some previous works implicitly minimize CVaR by only training on samples with top-k losses [e.g., 89]. Entropic Value-at-Risk (EVaR) is proposed as an upper bound of CVaR and VaR that could be more computationally efficient [9]. EVaR with a confidence level α $(0 < \alpha < 1)$ is defined as:

$$\text{EVaR}_X(1 - \alpha; \theta) := \min_{t \in \mathbb{R}^{>0}} \left\{ \frac{1}{t} \log \left(\frac{\mathbb{E}[e^{tf(X;\theta)}]}{\alpha} \right) \right\} = \min_{t \in \mathbb{R}^{>0}} \left\{ R_X(t; \theta) - \frac{1}{t} \log \alpha \right\}. \tag{7.62}$$

Similarly, for $\alpha \in \{\frac{k}{N}\}_{k\in[N]}$, the empirical variants of CVaR and EVaR are

$$\widetilde{\mathrm{CVaR}}(1-\alpha;\theta) := \min_{\gamma}\left\{\gamma + \frac{1}{\alpha}\frac{1}{N}\sum_{i\in[N]}[f(x_i;\theta)-\gamma]_+\right\},$$
$$\widetilde{\mathrm{EVaR}}(1-\alpha;\theta) := \min_{t\in\mathbb{R}^{>0}}\left\{\frac{1}{t}\log\left(\frac{\frac{1}{N}\sum_{i\in[N]}e^{tf(x_i;\theta)}}{\alpha}\right)\right\} = \min_{t\in\mathbb{R}^{>0}}\left\{\widetilde{R}(t;\theta) - \frac{1}{t}\log\alpha\right\}.$$

Notice that TERM objective appears as part of the objective in $\widetilde{\mathrm{EVaR}}$, and particularly optimizing $\widetilde{\mathrm{EVaR}}$ with respect to θ would be equivalent to solving TERM for some value of t implicitly defined through α (see Lemma 46 and Lemma 47 in the appendix).

It is known that $\mathrm{VaR}_X(1-\alpha;\theta) \leqslant \mathrm{CVaR}_X(1-\alpha;\theta) \leqslant \mathrm{EVaR}_X(1-\alpha;\theta)$ [9] which directly yields $\widetilde{\mathrm{VaR}}(1-\alpha;\theta) \leqslant \widetilde{\mathrm{CVaR}}(1-\alpha;\theta) \leqslant \widetilde{\mathrm{EVaR}}(1-\alpha;\theta)$. Meanwhile, to the best of our knowledge, it is not clear from existing works how entropic risk (or TERM) is related to VaR or EVaR. Next, based on TERM, we propose a new risk-averse objective Tilted Value-at-Risk, showing that it upper bounds VaR and lower bounds EVaR.

Definition 15 (Tilted Value-at-Risk (TiVaR))**.** *Let TiVaR for $\alpha \in (0,1]$ be defined as*

$$\mathit{TiVaR}_X(1-\alpha;\theta) := \min_{t\in\mathbb{R}}\left\{F_X(-\infty) + \frac{1}{t}\log\left[\frac{e^{(R_X(t;\theta)-F_X(-\infty))t} - (1-\alpha)}{\alpha}\right]_+\right\}. \tag{7.63}$$

Similarly, empirical TiVaR is defined for $\alpha \in (0,1)$,

$$\widetilde{\mathit{TiVaR}}\,(1-\alpha;\theta) := \min_{t\in\mathbb{R}}\left\{\widetilde{F}(-\infty) + \frac{1}{t}\log\left[\frac{e^{(\widetilde{R}(t;\theta)-\widetilde{F}(-\infty))t} - (1-\alpha)}{\alpha}\right]_+\right\}. \tag{7.64}$$

We note that TiVaR is not a coherent risk measure (see the work of Artzner [18], Artzner et al. [19] for definition of coherent risks), despite that it can be tighter than CVaR in some cases, as discussed in detail later. We next present our main result on relations between TiVaR, VaR, and EVaR.

Theorem 28. *For $\alpha \in (0,1]$ and any θ,*

$$\mathit{VaR}_X(1-\alpha;\theta) \leqslant \mathit{TiVaR}_X(1-\alpha;\theta) \leqslant \mathit{EVaR}_X(1-\alpha;\theta). \tag{7.65}$$

Similarly, for $\alpha \in \{\frac{k}{N}\}_{k\in[N]}$ and any θ,

$$\widetilde{\mathit{VaR}}\,(1-\alpha;\theta) \leqslant \widetilde{\mathit{TiVaR}}\,(1-\alpha;\theta) \leqslant \widetilde{\mathit{EVaR}}\,(1-\alpha;\theta)\,. \tag{7.66}$$

We defer the proof to Appendix 7.10, where the main steps include applying the new Chernoff bound variant (Theorem 27). Theorem 28 indicates that $\widetilde{\mathrm{TiVaR}}\,(1-\alpha;\theta)$ is a tighter approximation to $\widetilde{\mathrm{VaR}}\,(1-\alpha;\theta)$ than $\widetilde{\mathrm{EVaR}}\,(1-\alpha;\theta)$.

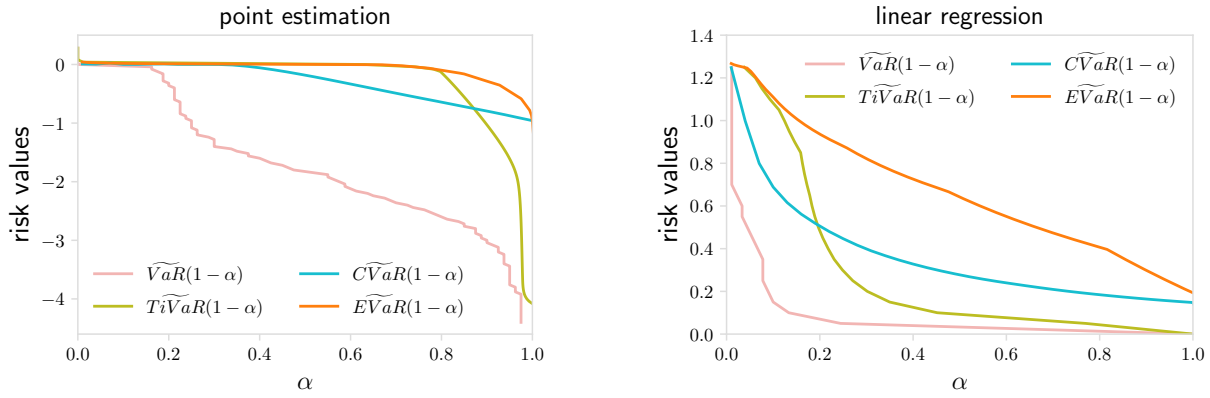


Figure 7.4: Comparing values of VaR, TiVaR, CVaR, and EVaR. $\widetilde{\text{VaR}}(1-\alpha) := \min_\theta \widetilde{\text{VaR}}(1-\alpha;\theta)$, and $\widetilde{\text{TiVaR}}(1-\alpha)$, $\widetilde{\text{CVaR}}(1-\alpha)$, and $\widetilde{\text{EVaR}}(1-\alpha)$ are defined in a similar way. From Theorem 28, we know $\widetilde{\text{VaR}}(1-\alpha;\theta) \leqslant \widetilde{\text{TiVaR}}(1-\alpha;\theta) \leqslant \widetilde{\text{EVaR}}(1-\alpha;\theta)$, which is also visualized here. Both CVaR and TiVaR values are between VaR and EVaR. TiVaR provides a tighter approximation to VaR than CVaR when α is closer to 1.

Comparing TiVaR and CVaR. In general, TiVaR and CVaR are not directly comparable, as both of them can be viewed as approximations to VaR and neither dominates the other, i.e., one risk can be tighter than the other depending on the quantile value, 1-α. In the regimes where α is a large value between some intermediate constant and 1, TiVaR provides a tighter approximation to VaR than CVaR. For instance, in the extreme case when $\alpha \to 1$, $\widetilde{\text{VaR}}$ will be close to the min-loss ($\min_{i\in[N]} f(x_i;\theta)$), while the value of $\widetilde{\text{CVaR}}$ is the mean of the losses ($\frac{1}{N}\sum_{i\in[N]} f(x_i;\theta)$). $\widetilde{\text{TiVaR}}$ reduces to the min-loss in this case. In other words, both $\widetilde{\text{VaR}}$ and $\widetilde{\text{TiVaR}}$ sweep the values between the min-loss and max-loss; whereas $\widetilde{\text{CVaR}}$ sweeps the values between the avg-loss and max-loss. We compare TiVaR with CVaR and other risks in Figure 7.4 on mean estimation and linear regression problems, and demonstrate that TiVaR is tighter than CVaR especially when α is close 1 (corresponding to robustness applications).[5]

We also note that there exist other risk-averse or risk-seeking formulations that focus on the upper or lower tail of losses, such as the mean-semideviation framework [145]. Mean-semideviation recovers a set of risk measures including mean-upper-semideviations and entropic mean-semideviation. Nevertheless, these risks usually cannot handle both fairness and robustness in a single formulation, and can incur more per-iteration gradient evaluations or worse convergence rates compared to vanilla ERM [105, 145, 334].

Finally, we draw connections between the above results and the k-loss, defined as the k-th smallest loss of N (i.e., 1-loss is the min-loss, N-loss is the max-loss, $(N-1)/2$-loss is

[5]While CVaR focuses on upper quantiles, one may explore 'inverse' CVaR to better approximate the lower quantiles. However, inverse CVaR, ranging from avg-loss to min-loss, is not a valid upper bound of VaR. Despite this, we empirically explore this approximation to solving VaR, among others, in Appendix 7.10.

the median-loss). Formally, let $R_{(k)}(\theta)$ be the k-th order statistic of the loss vector. Hence, $R_{(k)}$ is the k-th smallest loss, and particularly

$$R_{(1)}(\theta) = \check{R}(\theta), \quad R_{(N)}(\theta) = \widehat{R}(\theta). \tag{7.67}$$

Thus, for any $k \in [N]$, we define

$$R^*_{(k)} := \min_{\theta} R_{(k)}(\theta), \quad \theta^*(k) := \arg\min_{\theta} R_{(k)}(\theta). \tag{7.68}$$

Note that

$$R^*_{(1)} = \widetilde{F}(-\infty), \quad R^*_{(N)} = \widetilde{F}(+\infty). \tag{7.69}$$

While minimizing the k-loss is more desirable than ERM in many applications, the k-loss is non-smooth (and generally non-convex), and is challenging to solve for large-scale problems [134, 223]. TERM offers a good approximation to k-loss as well. Note that if we fix $\alpha = 1 - \frac{k}{N}$, minimizing k-loss is equivalent to minimizing γ where $\widetilde{Q}(\gamma;\theta) = \alpha$. Based on the bound of $\widetilde{\text{VaR}}$, we obtain a bound on k-loss:

Corollary 6. *For all $k \in \{2,\dots,N-1\}$, and all $t \in \mathbb{R}$:*

$$\begin{aligned} R_{(k)}(\theta) &\leqslant \min_{t} \left\{ \widetilde{F}(-\infty) + \frac{1}{t} \log \left[\frac{e^{(\widetilde{R}(t;\theta) - \widetilde{F}(-\infty))t} - \frac{k}{N}}{1 - \frac{k}{N}} \right]_+ \right\} \\ &\leqslant \min_{t \in \mathbb{R}^{>0}} \left\{ \widetilde{R}(t;\theta) - \frac{1}{t} \log \left(1 - \frac{k}{N}\right) \right\}. \end{aligned} \tag{7.70}$$

Proof. Note that

$$R_{(k)}(\theta) = \widetilde{\text{VaR}}\left(\frac{k}{N};\theta\right). \tag{7.71}$$

The proof completes by setting $\alpha = 1 - \frac{k}{N}$ in Eq. (7.64) and noting $\widetilde{VaR}(1-\alpha;\theta) \leqslant \widetilde{TiVaR}(1-\alpha;\theta) \leqslant \widetilde{EVaR}(1-\alpha;\theta)$. □

Corollary 6 optimizes over all $t \in \mathbb{R}$ over the upper bound of $R_{(k)}(\theta)$, which can be relaxed to searching over positive t's, as stated in Corollary 7 below.

Corollary 7. *For all $k \in \{2,\dots,N-1\}$, and all $t \in \mathbb{R}^{>0}$:*

$$R_{(k)}(\theta) \leqslant \widetilde{F}(-\infty) + \frac{1}{t} \log \left(\frac{e^{(\widetilde{R}(t;\theta) - \widetilde{F}(-\infty))t} - \frac{k}{N}}{1 - \frac{k}{N}} \right). \tag{7.72}$$

7.5 Solving TERM

In this section, we develop first-order batch (Section 7.5.1) and stochastic (Section 7.5.2) optimization methods for solving TERM, and rigorously analyze the effects that t has on the convergence of these methods.

Recall that in Section 7.2.2, we discuss the Lipschitzness, convexity, and smoothness properties of TERM. t-tilted loss remains strongly convex for $t > 0$, so long as the original loss function is strongly convex. On the other hand, for sufficiently large negative t, the t-tilted loss becomes non-convex. Hence, while the t-tilted solutions for positive t are unique, the objective may have multiple (spurious) local minima for negative t even if the original loss function is strongly convex. For negative t, we seek the solution for which the parametric set of t-tilted solutions obtained by sweeping $t \in \mathbb{R}$ (i.e., $\breve{\theta}(t)$ defined in Eq. (7.10)) remains continuous (as in Figure 7.1a-c and Figure 7.2). To this end, for negative t, we solve TERM by smoothly decreasing t from 0 observing that the solutions form a continuum in $\mathbb{R}^d$ empirically. Despite the non-convexity of TERM with $t < 0$, we find that this approach produces effective solutions to multiple real-world problems in Section 7.7. Additionally, as the objective remains smooth, it is still relatively efficient to solve. On the toy problem studied in Figure 7.2, we plot the convergence with t in Figure 7.5 below.

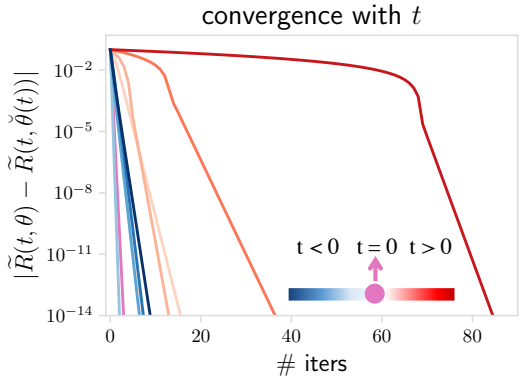


Figure 7.5: As $t \to +\infty$, the objective becomes less smooth in the vicinity of the final solution where smoothness can be measured by the upper bound of Hessian (see Lemma 24), hence suffering from slower convergence. For negative values of t, TERM converges fast due to the smoothness in the vicinity of solutions despite its non-convexity.

7.5.1 First-Order Batch Methods

TERM solver in the batch setting is summarized in Algorithm 13. The main steps include running gradient descent on $\widetilde{R}(t;\theta)$, which involve computing the tilted gradients (i.e., a weighted aggregation of individual gradients (Lemma 26)) of the objective. We also provide convergence results in Theorem 29–31 below for Algorithm 13.

Algorithm 13 Batch Non-Hierarchical TERM

Input: t, α, θ
for iter $= 0, \cdots, T-1$ **do**
  Compute the loss $f(x_i;\theta)$ and gradient $\nabla_\theta f(x_i;\theta)$ for all $i \in [N]$
  $\widetilde{R}(t;\theta) \leftarrow t$-tilted loss (7.2) on all $i \in [N]$
  $w_i(t;\theta) \leftarrow e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}$
  $\theta \leftarrow \theta - \frac{\alpha}{N}\sum_{i\in[N]} w_i(t;\theta)\nabla_\theta f(x_i;\theta)$
end for

Theorem 29 (Convergence of Algorithm 13 for strongly-convex problems)**.** *Under Assumption 9, there exist $\beta_{\max} \leqslant C_1 < \infty$ and $C_2 < \infty$ that do not depend on t such that for any $t \in \mathbb{R}^{>0}$, setting the step size $\alpha = \frac{1}{C_1+C_2 t}$, after k iterations:*

$$\widetilde{R}(t,\theta_k) - \widetilde{R}(t,\breve{\theta}(t)) \leqslant \left(1 - \frac{\beta_{\min}}{C_1 + C_2 t}\right)^k \left(\widetilde{R}(t,\theta_0) - \widetilde{R}(t,\breve{\theta}(t))\right). \tag{7.73}$$

Proof. First note that by Lemma 23, $\widetilde{R}(t,\theta)$ is $\beta_{\min}$-strongly convex for all $t \in \mathbb{R}^{>0}$. Next, by Lemma 24, there exist $C_1, C_2 < \infty$ such that $\widetilde{R}(t;\theta)$ has $(C_1 + C_2 t)$-Lipschitz gradients for all $t \in \mathbb{R}^{>0}$. The result follows directly from Karimi et al. [147, Theorem 1]. □

Note that under additional assumptions on L-Lipschitzness of $f(x;\theta)$, we can plug in the explicit smoothness constants established by Lowy and Razaviyayn [197, Lemma 5.3] to obtain explicit constants in the convergence rate, i.e., $C_1 = \beta_{\max}$ and $C_2 = L^2$. Theorem 29 indicates that solving TERM to a local optimum using gradient-based methods tends to be as efficient as traditional ERM for small-to-moderate values of t [133], which we corroborate via experiments on multiple real-world datasets in Section 7.7. This is in contrast to solving for the min-max solution, which would be similar to solving TERM as $t \to +\infty$ [157, 225, 227].

Theorem 30 (Convergence of Algorithm 13 for smooth problems satisfying PL conditions)**.** *Assume $f(x;\theta)$ is $\beta_{\max}$-smooth and (possibly) non-convex. Further assume $\sum_{i\in[N]} p_i f(x_i;\theta)$ is $\frac{\mu}{2}$-PL for any $\mathbf{p} \in \Delta_N$ where $\mathbf{p} := (p_1, \ldots, p_N)$. There exist $\beta_{\max} \leqslant C_1 < \infty$ and $C_2 < \infty$ that do not depend on t such that for any $t \in \mathbb{R}^{>0}$, setting the step size $\alpha = \frac{1}{C_1+C_2 t}$, after k iterations:*

$$\widetilde{R}(t,\theta_k) - \widetilde{R}(t,\breve{\theta}(t)) \leqslant \left(1 - \frac{\mu}{C_1 + C_2 t}\right)^k \left(\widetilde{R}(t,\theta_0) - \widetilde{R}(t,\breve{\theta}(t))\right), \tag{7.74}$$

Proof. If $\sum_{i\in[N]} p_i f(x_i;\theta)$ is μ-PL for any $\mathbf{p} \in \Delta_N$, then $\widetilde{R}(t;\theta)$ is μ-PL [232]. $\widetilde{R}(t;\theta)$ is $\beta_{\max}$ smooth for $t < 0$ and its smoothness parameter scales linearly with t for $t > 0$, following the same proof as Lemma 24. □

Theorem 30 applies to both convex and non-convex smooth functions satisfying PL conditions. Again, here we can plug in explicit smoothness parameter [197, Lemma 5.3] if $f(x;\theta)$ is Lipschitz. We next state results without the PL condition assumption for completeness.

Theorem 31 (Convergence of Algorithm 13 for non-convex smooth problems)**.** *Assume $f(x;\theta)$ is $\beta_{\max}$-smooth and (possibly) non-convex. Setting the step size $\alpha = \frac{1}{\beta(t)}$, after K iterations, we have:*

$$\frac{1}{K}\sum_{k=0}^{K-1} \|\nabla \widetilde{R}(t,\theta_k)\|^2 \leqslant \frac{2\beta(t)(\widetilde{R}(t,\theta_0) - \widetilde{R}(t,\breve{\theta}(t)))}{K}, \tag{7.75}$$

where for $t \in \mathbb{R}^{>0}$, $\beta(t) = C_1 + Ct$ where C_1, C_2 are independent of t and $\beta_{\max} \leqslant C_1 < \infty, C_2 < \infty$, and for $t \in \mathbb{R}^{-}$, $\beta(t) = \beta_{\max}$.

Theorem 31 also covers the case of convex $f(x;\theta)$ with $t < 0$. We note that for non-convex problems, when $t < 0$, the convergence rate is independent of t under our assumptions. We also observe this on a toy problem in Figure 7.5. In all applications we studied in Section 7.7 with negative t's, TERM runs the same number iterations as those of ERM.

7.5.2 First-Order Stochastic Methods

To obtain unbiased stochastic gradients, we need to have access to the normalization weights for each sample (i.e., $\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}$), which is often intractable to compute for large-scale problems. Hence, we use $\widetilde{\widetilde{R}}_t$, a term that incorporates stochastic dynamics, to estimate the tilted objective $\widetilde{R}_t := \widetilde{R}(t;\theta)$, which is used for normalizing the weights as in (7.25). In particular, we do not use a trivial linear averaging of the current estimate and the history to update $\widetilde{\widetilde{R}}_t$. Instead, we use a tilted averaging to ensure an unbiased estimator (if θ is not being updated).

On the other hand, the TERM objective can be viewed as a composition of functions $\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}$ and $\frac{1}{t}\log(\cdot)$, and could be optimized based on previous stochastic compositional optimization techniques [e.g., 99, 232, 233, 292, 293]. Similar to Wang et al. [293], we maintain two sequences (in our context, the model θ and the objective estimate $\widetilde{\widetilde{R}}_t$) throughout the optimization process. This (non-hierarchical) stochastic algorithm is summarized in Algorithm 14 below.

For the purpose of analysis, we sample two independent mini-batches to obtain the gradient of the original loss functions $\nabla_\theta f(x;\theta)$ and update $\widetilde{\widetilde{R}}_t$, respectively (described in Algorithm 17 for completeness). As we will see in Theorem 32, the additional randomness allows us to achieve better convergence rates compared with the algorithm proposed in Wang et al. [293] instantiated to our objective. Our rate of this simple algorithm matches the rate of more complicated ones [232], and developing optimal optimization procedures is out of the scope of this work. Empirically, we observe that sampling two mini-batches yield similar performance as using the same mini-batch to query the individual losses and the weights (Figure 7.17 in Appendix 7.11.2). Therefore, we employ the cheaper variant of just involving one mini-batch (Algorithm 14) in the corresponding experiments.

Algorithm 14 Stochastic Non-Hierarchical TERM

Input: θ, $\widetilde{\widetilde{R}}_t = \frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}\right)$, t, α, λ
for iter $= 0, \cdots, T-1$ **do**
  sample a minibatch B uniformly at random from $[N]$
  compute the loss $f(x;\theta)$ and gradient $\nabla_\theta f(x;\theta)$ for all $x \in B$
  $\widetilde{R}_{B,t} \leftarrow t$-tilted loss (7.2) on minibatch B
  $\widetilde{\widetilde{R}}_t \leftarrow \frac{1}{t}\log\left((1-\lambda)e^{t\widetilde{\widetilde{R}}_t} + \lambda e^{t\widetilde{R}_{B,t}}\right)$
  $w_{t,x} \leftarrow e^{tf(x;\theta) - t\widetilde{\widetilde{R}}_t}$
  $\theta \leftarrow \theta - \frac{\alpha}{|B|}\sum_{x\in B} w_{t,x}\nabla_\theta f(x;\theta)$
end for

The stochastic algorithm developed here requires roughly the same time/space complexity as mini-batch SGD, and thus scales similarly for large-scale problems. It can also help mitigate the potential numerical issues in implementation caused by the exponential tilting operator. We find that these methods perform well empirically on a variety of tasks (Section 7.7).

Theorem 32 (Convergence of Algorithm 17 for strongly-convex problems)**.** *Assume $f : \mathcal{X} \times \Theta \to [\widetilde{F}_{\min}, \widetilde{F}_{\max}]$ is L-Lipschitz in θ, i.e., $\widetilde{F}_{\min} \leqslant f(x;\theta) \leqslant \widetilde{F}_{\max}$,[6] and $|f(x;\theta_i) - f(x;\theta_j)| \leqslant L\|\theta_i - \theta_j\|$ for $x \in \mathcal{X}$ and $\theta_i, \theta_j \in \Theta \subseteq \mathbb{R}^d$. Assume $\widetilde{R}(t;\theta)$ has compact domain θ. Assume $\widetilde{R}(t;\theta)$ is μ-strongly convex (Assumption 9) with uniformly bounded stochastic gradient, i.e., $\|\nabla\widetilde{R}(x_i;\theta)\| := \left\|\frac{e^{tf(x_i;\theta)}}{e^{t\widetilde{R}(t;\theta)}}\nabla f(x_i;\theta)\right\| \leqslant B$ for $\theta \in \mathbb{R}^d$ and $i \in [N]$. Denote $k_t := \arg\max_k \left(k < \frac{2e}{\mu} + \frac{etLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{\mu k}\right)$. Assume the batch size is 1. For $k \geqslant k_t$,*

$$\mathbb{E}[\|\theta_{k+1} - \theta^*\|^2] \leqslant \frac{V_t}{k+1}, \tag{7.76}$$

where

$$\theta^* := \breve{\theta}(t), \quad V_t = \max\left\{k_t\mathbb{E}[\|\theta_{k_t} - \theta^*\|^2], \frac{4B^2e^{2+2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{\mu^2}\right\}, \tag{7.77}$$

and

$$\mathbb{E}[\|\theta_{k_t} - \theta^*\|^2] \leqslant \max\left\{\mathbb{E}[\|\theta_1 - \theta^*\|^2], \frac{B^2e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})+1}}{\mu(1 + tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})})}\right\}. \tag{7.78}$$

Our assumptions are standard compared with those in related literature [233, 293]. The uniformly bounded stochastic gradient of $\widetilde{R}(t;\theta)$ assumption can be satisfied by the bounded gradient of $f(x_i;\theta)$, which can be a limiting condition but has appeared

[6] For notation consistency between the max-loss and min-loss for any sample and any iteration, we use $\widetilde{F}_{\min}$ to denote the lower bound of $f(x_i;\theta_k)$. We note that $\widetilde{F}_{\min} = \widetilde{F}(-\infty)$ defined in Definition 7.11.

in previous works on stochastic compositional optimization [233, 292]. If the objectives are coercive, which typically holds in practice [28], Algorithm 14 will have bounded iterates and thus the compact domain assumption would hold. We defer full proofs to Appendix 7.11.2. The main steps involve bounding the expected estimation error $\mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)}]$ conditioning on the previous iterates $\{\theta_1, \ldots, \theta_k\}$.

Discussions. The theorem indicates that Algorithm 14 starts to make progress after k_t iterations, with convergence rate $O(e^{2t}/k)$. Both k_t and V_k could scale exponentially with t in the worst-case analysis, but it does not completely reflect the dependence of Algorithm 14 on t for modest values of t. Empirically, we observe that the stochastic TERM solver with moderate values of t can converge faster compared with stochastic min-max solvers, which has a rate of $1/\sqrt{k}$ for strongly convex problems [167]. This leaves open for future work understanding the exact scaling of the convergence rate of stochastic TERM as $t \to \infty$.

Next, we present convergence results on non-convex smooth problems, without and with the assumptions of PL-conditions. We defer all proofs to Appendix 7.11.2.

Theorem 33 (Convergence of Algorithm 17 for non-convex smooth problems)**.** *Assume $f : \mathcal{X} \times \Theta \to [\widetilde{F}_{\min}, \widetilde{F}_{\max}]$ is L-Lipschitz in θ, i.e., $\widetilde{F}_{\min} \leqslant f(x;\theta) \leqslant \widetilde{F}_{\max}$, and $|f(x;\theta_i) - f(x;\theta_j)| \leqslant L\|\theta_i - \theta_j\|$ for $x \in \mathcal{X}$ and $\theta_i, \theta_j \in \Theta \subseteq \mathbb{R}^d$. Assume $\widetilde{R}(t;\theta)$ is β-smooth with uniformly bounded stochastic gradient, i.e., $\|\nabla \widetilde{R}(x_i;\theta)\| \leqslant B$ for $\theta \in \mathbb{R}^d$ and $i \in [N]$. Assume the batch size is 1. Denote $k_t := \left\lceil \frac{2(\widetilde{F}_{\max} - \widetilde{F}_{\min})t^2L^2}{\beta e^2} \right\rceil$, then for $k \geqslant k_t$,*

$$\frac{1}{K}\sum_{k=k_t}^{K} \mathbb{E}[\|\nabla \widetilde{R}(t;\theta_k)\|^2] \leqslant \sqrt{8}Be^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})+1}\sqrt{\frac{\beta(\widetilde{F}_{\max} - \widetilde{F}_{\min})}{K}}. \tag{7.79}$$

Theorem 34 (Convergence of Algorithm 17 for non-convex smooth problems with PL conditions)**.** *Let the assumptions in Theorem 33 hold. Further assume that $\sum_{i\in[N]} p_i f(x_i;\theta)$ satisfies $\frac{\mu}{2}$-PL conditions for any $\mathbf{p} \in \Delta_N$ where $\mathbf{p} := (p_1, \ldots, p_N)$. Assume the batch size is 1. Denote $k_t := \arg\max_k \left(k < \frac{4e}{\mu} + \frac{4etLBE^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{\mu k}\right)$, then for $t \in \mathbb{R}^{>0}$ and $k \geqslant k_t$,*

$$\mathbb{E}[\widetilde{R}(t;\theta_{k+1}) - \widetilde{R}(t;\breve{\theta}(t))] \leqslant \frac{V_t}{k+1}, \tag{7.80}$$

where

$$V_t = \max\left\{k_t\mathbb{E}[\widetilde{R}(t;\theta_{k_t}) - \widetilde{R}(t;\breve{\theta}(t))], \frac{8\beta B^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})+2}}{\mu^2}\right\}. \tag{7.81}$$

7.6 TERM Extended: Hierarchical Multi-Objective Tilting

We consider an extension of TERM that can be used to address practical applications requiring multiple objectives, e.g., simultaneously achieving robustness to noisy data *and* ensuring fair performance across subgroups. Existing approaches typically aim to address such problems in isolation. To handle multiple objectives with TERM, let each sample x be associated with a group $g \in [G]$, i.e., $x \in g$. These groups could be related to the labels (e.g., classes in a classification task), or may depend only on features. For any $t, \tau \in \mathbb{R}$, we define multi-objective TERM as:

$$\widetilde{J}(t, \tau; \theta) := \frac{1}{t} \log \left(\frac{1}{N} \sum_{g \in [G]} |g| e^{t \widetilde{R}_g(\tau; \theta)} \right), \tag{7.82}$$

where

$$\widetilde{R}_g(\tau; \theta) := \frac{1}{\tau} \log \left(\frac{1}{|g|} \sum_{x \in g} e^{\tau f(x; \theta)} \right), \tag{7.83}$$

and $|g|$ is the size of group g. We evaluate the gradient of the hierarchical multi-objective tilt objective in Lemma 31 below.

Lemma 31 (Hierarchical multi-objective tilted gradient)**.** *Under Assumption 8,*

$$\nabla_\theta \widetilde{J}(t, \tau; \theta) = \sum_{g \in [G]} \sum_{x \in g} w_{g,x}(t, \tau; \theta) \nabla_\theta f(x; \theta), \tag{7.84}$$

where

$$w_{g,x}(t, \tau; \theta) := \frac{\left(\frac{1}{|g|} \sum_{y \in g} e^{\tau f(y; \theta)} \right)^{\left(\frac{t}{\tau} - 1\right)}}{\sum_{g' \in [G]} |g'| \left(\frac{1}{|g'|} \sum_{y \in g'} e^{\tau f(y; \theta)} \right)^{\frac{t}{\tau}}} \quad e^{\tau f(x; \theta)}. \tag{7.85}$$

Similar to the tilted gradient (7.25), Lemma 31 indicates that the multi-objective tilted gradient is a weighted sum of the gradients, making TERM similarly efficient to solve. Multi-objective TERM recovers sample-level TERM as a special case for $\tau = t$ (Lemma 32), and reduces to group-level TERM with $\tau \to 0$.

Lemma 32 (Sample-level TERM is a special case of hierarchical multi-objective TERM)**.** *Under Assumption 8, hierarchical multi-objective TERM recovers TERM as a special case for $t = \tau$. That is*

$$\widetilde{J}(t, t; \theta) = \widetilde{R}(t; \theta). \tag{7.86}$$

Proof. The proof is completed by noticing that setting $t = \tau$ in (7.85) recovers the original sample-level tilted gradient. □

Note that all properties discussed in Section 7.2 carry over to group-level TERM. We validate the effectiveness of hierarchical tilting empirically in Section 7.7.3, where we show that TERM can significantly outperform baselines to handle class imbalance *and* noisy outliers simultaneously, while underperforming a much more complicated method in their setup. Note that hierarchical tilting could be extended to hierarchies of greater depths (than two) to simultaneously handle more than two objectives at the cost of one extra tilting hyperparameter per each additional optimization objective. For instance, we state the multi-objective tilting for a hierarchy of depth three in Appendix 7.11.1.

7.6.1 Solving Hierarchical TERM

To solve hierarchical TERM in the batch setting, we can directly use gradient-based methods with tilted gradients defined for the hierarchical objective in Lemma 31. Note that Batch hierarchical TERM with $t = \tau$ reduces to solving the sample-level tilted objective (7.2). We summarize this method in Algorithm 15.

Algorithm 15 Batch Hierarchical TERM

Input: t, τ, α
for iter $= 0, \cdots, T-1$ **do**
  for $g \in [G]$ **do**
    compute the loss $f(x;\theta)$ and gradient $\nabla_\theta f(x;\theta)$ for all $x \in g$
    $\widetilde{R}_{g,\tau} \leftarrow \tau$-tilted loss (7.82) on group g
    $\nabla_\theta \widetilde{R}_{g,\tau} \leftarrow \frac{1}{|g|} \sum_{x \in g} e^{\tau f(x;\theta) - \tau \widetilde{R}_{g,\tau}} \nabla_\theta f(x;\theta)$
  end for
  $\widetilde{J}_{t,\tau} \leftarrow \frac{1}{t} \log \left(\frac{1}{N} \sum_{g \in [G]} |g| e^{t \widetilde{R}_g(\tau;\theta)} \right)$
  $w_{t,\tau,g} \leftarrow |g| e^{t \widetilde{R}_{\tau,g} - t \widetilde{J}_{t,\tau}}$
  $\theta \leftarrow \theta - \frac{\alpha}{N} \sum_{g \in [G]} w_{t,\tau,g} \nabla_\theta \widetilde{R}_{g,\tau}$
end for

We next discuss stochastic solvers for hierarchical multi-objective tilting. We extend Algorithm 14 to the multi-objective setting, presented in Algorithm 16. At a high level, at each iteration, group-level tilting is addressed by choosing a group based on the tilted weight vector. Sample-level tilting is then incorporated by re-weighting the samples in a uniformly drawn mini-batch. Similarly, we estimate the tilted objective $\widetilde{R}_{g,\tau}$ for each group g via a tilted average of the current estimate and the history. While we sample the group from which we draw the minibatch, for small number of groups, one might want to draw one minibatch per each group and weight the resulting gradients accordingly.

Group-level tilting can be recovered from Algorithm 15 and 16 by setting the inner-level tilt parameter $\tau = 0$. We apply TERM to a variety of machine learning problems; for clarity, we summarize the applications and their corresponding algorithms in Table 7.10 in the appendix.

Algorithm 16 Stochastic Hierarchical TERM

Input: $t, \tau, \alpha, \lambda, \widetilde{\widetilde{R}}_{g,\tau} = 0 \; \forall g \in [G]$
for $t = 0, \cdots, T-1$ **do**
  sample g on $[G]$ from a Gumbel-Softmax distribution with logits $\widetilde{\widetilde{R}}_{g,\tau} + \frac{1}{t}\log|g|$ and temperature $\frac{1}{t}$
  sample minibatch B uniformly at random within group g
  compute the loss $f(x;\theta)$ and gradient $\nabla_\theta f(x;\theta)$ for all $x \in B$
  $\widetilde{R}_{B,\tau} \leftarrow \tau$-tilted loss (7.2) on minibatch B
  $\widetilde{\widetilde{R}}_{g,\tau} \leftarrow \frac{1}{\tau}\log\left((1-\lambda)e^{\tau\widetilde{\widetilde{R}}_{g,\tau}} + \lambda e^{\tau\widetilde{R}_{B,\tau}}\right)$
  $w_{\tau,x} \leftarrow e^{\tau f(x;\theta) - \tau\widetilde{\widetilde{R}}_{g,\tau}}$
  $\theta \leftarrow \theta - \frac{\alpha}{|B|}\sum_{x\in B} w_{\tau,x}\nabla_\theta f(x;\theta)$
end for

7.7 TERM in Practice: Use Cases

We now showcase the flexibility, wide applicability, and competitive performance of the TERM framework through empirical results on a variety of real-world problems such as handling outliers (Section 7.7.1), ensuring fairness and improving generalization (Section 7.7.2), and addressing compound issues (Section 7.7.3). Despite the relatively straightforward modification TERM makes to traditional ERM, we show that t-tilted losses not only outperform ERM, but either outperform or are competitive with state-of-the-art, problem-specific tailored baselines on a wide range of applications. We provide implementation details in Appendix 7.12.2. All code, datasets, and experiments are publicly available at `github.com/litian96/TERM`. The applications explored are summarized in Table 7.1 below.

Table 7.1: Summary of TERM applications.

Applications		Sections
Mitigating noisy outliers ($t < 0$)	Robust regression	Sec. 7.7.1.1
	Robust classification	Sec. 7.7.1.2
	Low-quality annotators	Sec. 7.7.1.3
Fairness and generalization ($t > 0$)	Fair PCA	Sec. 7.7.2.1
	Fair federated learning	Sec. 7.7.2.2
	Fair meta-learning	Sec. 7.7.2.3
	Handling class imbalance	Sec. 7.7.2.4
	Improving generalization	Sec. 7.7.2.5
Hierarchical multi-objective tilting	Class imbalance and random noise	Sec. 7.7.3.1
	Class imbalance and adversarial noise	Sec. 7.7.3.2

Choosing t. In applications when we consider tradeoffs between different objectives (e.g., fair meta-learning and federated learning), we perform a grid search over t from $\{0.1, 1, 2, 5, 10, 50, 100, 200\}$ on the validation set and pick the one with the best fairness performance while not degrading mean performance. When there is not a single t dominating other values (e.g., fair PCA), we report results under different values of t. In our initial robust regression experiments, we find that the performance is robust to various t's, and we thus use a fixed $t = -2$ for all experiments involving negative t (Section 7.7.1 and Section 7.7.3). For all values of t tested, the number of iterations required to solve TERM is within $2\times$ that of standard ERM, with the same per-iteration complexity.

7.7.1 Mitigating Noisy Outliers ($t < 0$)

We begin by investigating TERM's ability to find robust solutions that reduce the effect of noisy outliers. We note that we specifically focus on the setting of 'robustness' involving random additive noise; the applicability of TERM to more adversarial forms of robustness would be an interesting direction of future work. We do not compare with approaches that require additional clean validation data [e.g., 115, 238, 244, 284], as such data can be costly to obtain in practice.

7.7.1.1 Robust Regression

Label Noise. We first consider a regression task with noise corrupted targets, where we aim to minimize the root mean square error (RMSE) on samples from the Drug Discovery dataset [71, 224]. The task is to predict the bioactivities given a set of chemical compounds. We compare against linear regression with an L_2 loss, which we view as the 'standard' ERM solution for regression, as well as with losses commonly used to mitigate outliers—the L_1 loss and Huber loss [125]. We also compare with consistent robust regression (CRR) [31] and STIR [217], recent state-of-the-art methods specifically designed for label noise in robust regression. In this particular problem, TERM is equivalent to exponential squared loss, studied in [295]. We apply TERM at the sample level with an L_2 loss, and generate noisy outliers by assigning random targets drawn from $\mathcal{N}(5, 5)$ on a fraction of the samples.

In Table 7.2, we report RMSE on clean test data for each objective and under different noise levels. We also present the performance of an oracle method (Genie ERM) which has access to all of the clean data samples with the noisy samples removed. *Note that Genie ERM is not a practical algorithm and is solely presented to set the expected performance limit in the noisy setting*. The results indicate that TERM is competitive with baselines on the 20% noise level, and achieves better robustness with moderate-to-extreme noise. We observe similar trends in scenarios involving both noisy features and targets (Appendix 7.12.1). CRR tends to run slowly as it scales cubicly with the number of dimensions [31], while solving TERM is roughly as efficient as ERM.

Table 7.2: TERM is competitive with robust *regression* baselines, particularly in high noise regimes.

objectives	**test RMSE** (`Drug Discovery`)		
	20% noise	40% noise	80% noise
ERM	1.87 (.05)	2.83 (.06)	4.74 (.06)
L_1	**1.15** (.07)	1.70 (.12)	4.78 (.08)
Huber [125]	**1.16** (.07)	1.78 (.11)	4.74 (.07)
STIR [217]	**1.16** (.07)	1.75 (.12)	4.74 (.06)
CRR [31]	**1.10** (.07)	1.51 (.08)	4.07 (.06)
TERM	**1.08** (.05)	**1.10** (.04)	**1.68** (.03)
Genie ERM	1.02 (.04)	1.07 (.04)	1.04 (.03)

Label and Feature Noise. Here, we present results involving both feature noise and target noise. We investigate the performance of TERM on two datasets (cal-housing [226] and abalone [74]) used in Yu et al. [314]. Both datasets have features with 8 dimensions. We generate noisy samples following the setup in Yu et al. [314]—sampling 100 training samples, and randomly corrupting 5% of them by multiplying their features by 100 and multiply their targets by 10,000. From Table 7.3 below, we see that TERM significantly outperforms the baseline objectives in the noisy regime on both datasets.

Table 7.3: An alternative noise setup involving both feature and label noise. Similarly, TERM with $t = -2$ significantly outperforms several baseline objectives for noisy outlier mitigation.

objectives	**test RMSE** (`cal-housing`)		**test RMSE** (`abalone`)	
	clean	noisy	clean	noisy
ERM	0.766 (0.023)	239 (9)	2.444 (0.105)	1013 (72)
L_1	0.759 (0.019)	139 (11)	2.435 (0.021)	1008 (117)
Huber [125]	0.762 (0.009)	163 (7)	2.449 (0.018)	922 (45)
CRR [31]	0.766 (0.024)	245 (8)	2.444 (0.021)	986 (146)
TERM	0.745 (0.007)	**0.753** (0.016)	2.477 (0.041)	**2.449** (0.028)
Genie ERM	0.766 (0.023)	0.766 (0.028)	2.444 (0.105)	2.450 (0.109)

Unstructured Random v.s. Adversarial Noise. As a word of caution, we note that the experiments thus far have focused on random noise. This makes it possible for the methods to find the underlying structure of clean data even if the majority of the samples are noisy outliers. To gain more intuition on these cases, we generate synthetic two-dimensional data points and test the performance of TERM under 0%, 20%, 40%, and 80% noise for linear regression. TERM with $t = -2$ performs well in all noise levels (Figure 7.6 and 7.7). However, as one might expect, TERM with negative t's could potentially overfit to outliers if they are constructed in an adversarial way. In the

examples shown in Figure 7.8, under 40% noise and 80% noise, TERM has a high error measured on the clean data (green dots).

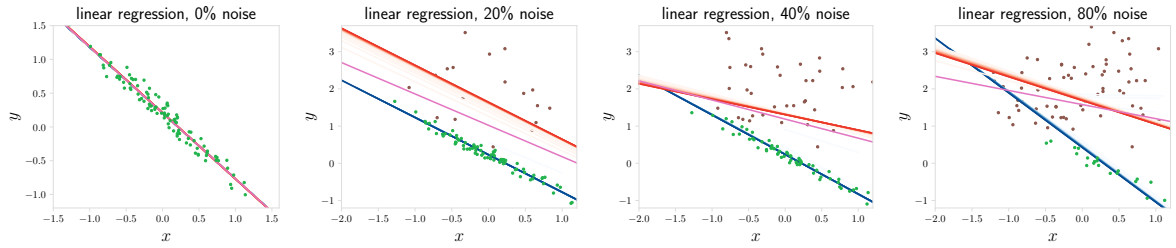


Figure 7.6: Robust regression on synthetic data with random noise where the mean of the noisy samples is different from that of clean ones. TERM with negative t's (blue, $t = -2$) can fit structured clean data at all noise levels, while ERM (purple) and TERM with positive t's (red) overfit to corrupted data. We color inliers in green and outliers in brown for visualization.

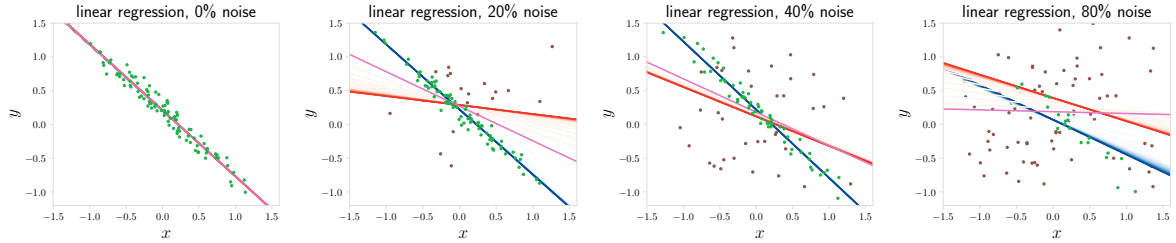


Figure 7.7: In the presence of random noise with the same mean as that of clean data, TERM with negative t's (blue) can still surpass outliers in all cases, while ERM (purple) and TERM with positive t's (red) overfit to corrupted data. While the performance drops for 80% noise, TERM can still learn useful information, and achieves much lower error than ERM.

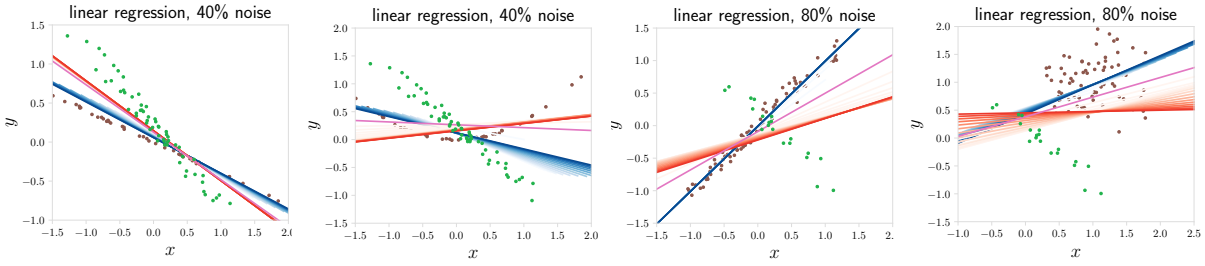


Figure 7.8: TERM with negative t's (blue) cannot fit clean data if the noisy samples (brown) are adversarial or structured in a manner that differs substantially from the underlying true distribution.

7.7.1.2 Robust Classification

Deep neural networks can easily overfit to corrupted labels [e.g., 319]. While the theoretical properties we study for TERM (Section 7.2) do not directly cover objectives with neural network function approximations, we show that TERM can be applied empirically to DNNs to achieve robustness to noisy training labels. MentorNet [130] is a popular method in this setting, which learns to assign weights to samples based on feedback from a student net. Following the setup in Jiang et al. [130], we explore classification on CIFAR10 [158] when a fraction of the training labels are corrupted with uniform noise—comparing TERM with ERM and several state-of-the-art approaches [158, 159, 238, 326]. As shown in Table 7.4, TERM performs competitively with 20% noise, and outperforms all baselines in the high noise regimes. We use MentorNet-PD as a baseline since it does not require clean validation data. In Appendix 7.12.1, we show that TERM also matches the performance of MentorNet-DD, which requires clean validation data. To help reason about the performance of TERM, we also explore a simpler, two-dimensional logistic regression problem in Figure 7.19, Appendix 7.12.1, finding that TERM with $t{=}{-}2$ is similarly robust across the considered noise regimes.

Table 7.4: TERM is competitive with robust *classification* baselines, and is superior in high noise regimes.

objectives	**test accuracy** (`CIFAR10`, Inception)		
	20% noise	40% noise	80% noise
ERM	0.775 (.004)	0.719 (.004)	0.284 (.004)
RandomRect [238]	0.744 (.004)	0.699 (.005)	0.384 (.005)
SelfPaced [159]	0.784 (.004)	0.733 (.004)	0.272 (.004)
MentorNet-PD [130]	0.798 (.004)	0.731 (.004)	0.312 (.005)
GCE [326]	**0.805** (.004)	0.750 (.004)	0.433 (.005)
TERM	0.795 (.004)	**0.768** (.004)	**0.455** (.005)
Genie ERM	0.828 (.004)	0.820 (.004)	0.792 (.004)

7.7.1.3 Low-Quality Annotators

It is not uncommon for practitioners to obtain human-labeled data for their learning tasks from crowd-sourcing platforms. However, these labels are usually noisy in part due to the varying quality of the human annotators. Given a collection of labeled samples from crowd-workers, we aim to learn statistical models that are robust to the potentially low-quality annotators. As a case study, following the setup of [152], we take the CIFAR-10 dataset and simulate 100 annotators where 20 of them are *hammers* (i.e., always correct) and 80 of them are *spammers* (i.e., assigning labels uniformly at random). We apply TERM at the annotator group level in (7.82), which is equivalent to assigning annotator-level weights based on the aggregate value of their loss. As shown in Figure 7.9, TERM is able

to achieve the test accuracy limit set by *Genie ERM*, i.e., *the ideal performance obtained by completely removing the known outliers*. We note in particular that the accuracy reported by [152] (0.777) is lower than TERM (0.825) in the same setup, even though their approach is a two-pass algorithm requiring at least to double the training time. We provide full empirical details and investigate additional noisy annotator scenarios in Appendix 7.12.1.

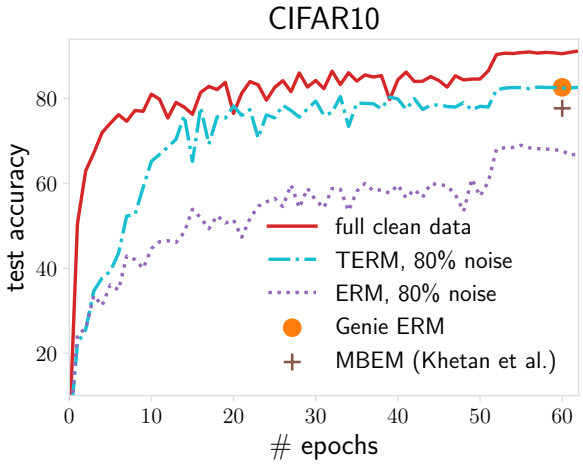


Figure 7.9: TERM (t=−2) completely removes the impact of noisy annotators, reaching the performance limit set by Genie ERM.

7.7.2 Fairness and Generalization ($t > 0$)

In this section, we show that positive values of t in TERM can help promote fairness via learning fair representations and enforcing fairness during optimization, and offer variance reduction for better generalization.

7.7.2.1 Fair Principal Component Analysis (PCA)

We explore the flexibility of TERM in learning fair representations using PCA. In fair PCA, the goal is to learn low-dimensional representations which are fair to all considered subgroups (e.g., yielding similar reconstruction errors) [146, 247, 281]. Despite the non-convexity of the fair PCA problem, we apply TERM to this task, referring to the resulting objective as TERM-PCA. We tilt the same loss function as in Samadi et al. [247]: $f(X;U) = \frac{1}{|X|}\left(\|X - XUU^\top\|_F^2 - \|X - \hat{X}\|_F^2\right)$, where $X \in \mathbb{R}^{n\times d}$ is a subset (group) of data, $U \in \mathbb{R}^{d\times r}$ is the current projection, and $\hat{X} \in \mathbb{R}^{n\times d}$ is the optimal rank-r approximation of X. Instead of solving a more complex min-max problem using semi-definite programming as in Samadi et al. [247], which scales poorly with problem dimension, we apply gradient-based methods, re-weighting the gradients at each iteration based on the loss on each group. In Figure 7.10, we plot the aggregate loss for two groups (high vs. low education) in the Default Credit dataset [308] for different target dimensions r. By varying t, we

achieve varying degrees of performance improvement on different groups—TERM ($t = 200$) recovers the min-max results of [247] by forcing the losses on both groups to be (almost) identical, while TERM ($t = 10$) offers the flexibility of reducing the performance gap less aggressively. We also provide convergence plots for different values of t in this application (Figure 7.11), and observe slower convergence for larger values of t, which is consistent with our analyses in Section 7.2 and 7.5. However, we do not observe exponential dependence on t from the convergence curves, which suggest that the theoretical dependence on t in convergence proofs for the solvers may be an artifact of our proof techniques, and might possibly be further improved by other analysis techniques for typical practical use cases.

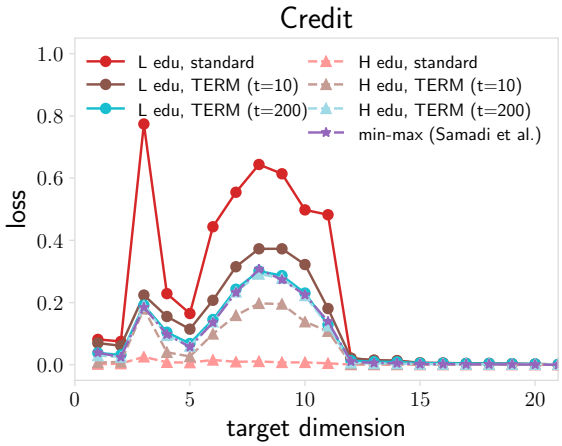


Figure 7.10: TERM-PCA flexibly trades the performance on the high (H) edu group for the performance on the low (L) edu group.

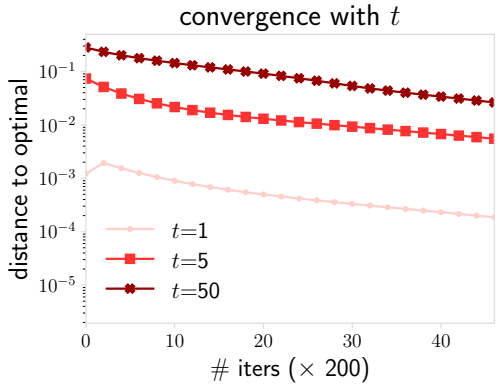


Figure 7.11: Convergence of TERM with respect to t in fair PCA (target dimension=7). We tune optimal learning rates separately for each t. As t increases, the convergence becomes slower, which validates our analyses in Section 7.2 and 7.5.

7.7.2.2 Fair Federated Learning

Federated learning involves learning statistical models across massively distributed networks of remote devices or isolated organizations [178, 208]. Ensuring fair (i.e., uniform) performance distribution across the devices is a major concern in federated settings [182, 215], as using current approaches for federated learning (FedAvg [208]) may result in highly variable performance across the network. Li et al. [182] consider solving an alternate objective for federated learning, called q-FFL, to dynamically emphasize the worst-performing devices, which is conceptually similar to the goal of TERM, though it is applied specifically to the problem of federated learning and limited to the case of positive t. Here, we compare TERM with q-FFL in their setup on the vehicle dataset [75] consisting of data collected from 23 distributed sensors (hence 23 devices). We tilt the L_2 regularized linear SVM objective at the device level. At each communication round, we re-weight the accumulated local model updates from each selected device based

on the weights estimated via Algorithm 16. From Figure 7.12, we see that similar to q-FFL, TERM ($t = 0.1$) can also significantly promote the accuracy on the worst device while maintaining the overall performance. The statistics of the accuracy distribution are reported in Table 7.5 below.

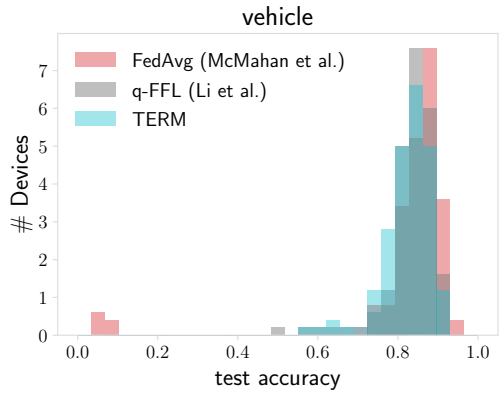


Figure 7.12: TERM FL ($t = 0.1$) significantly increases the accuracy on the worst-performing device (similar to q-FFL) while obtaining a similar average accuracy.

Table 7.5: Both q-FFL and TERM can encourage more uniform accuracy distributions across the devices in federated networks while maintaining similar average performance. Numbers in the parentheses correspond to the standard error of each metric across 5 runs.

objectives	**test accuracy**		
	average	worst 10%	stdev
FedAvg	0.853 (.078)	0.421 (.007)	0.173 (.001)
q-FFL ($q = 5$)	0.862 (.029)	**0.704** (.033)	**0.064** (.005)
TERM ($t = 0.1$)	0.853 (.027)	**0.707** (.009)	**0.061** (.003)

7.7.2.3 Fair Meta-Learning

Meta-learning aims to learn a shared initialization across all tasks such that the initialization can quickly adapt to unseen tasks (i.e., meta-testing tasks) using a few samples. In practice, the resulting performance across meta-testing tasks can vary due to different data distributions associated with these tasks. One of the popular meta-learning methods is MAML [92], whose objective is to minimize the sum of empirical losses across tasks $\{\mathcal{T}_i\}$ generated from $p(\mathcal{T})$ after one step of adaptation, i.e., $\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} f\left(\mathcal{T}_i; \theta - \alpha \nabla_\theta f(\mathcal{T}_i; \theta)\right)$. Previous works have proposed a min-max variant of MAML to encourage a more fair (uniform) performance distribution by optimizing the worst meta-training task called TR-MAML [57]. We apply TERM to MAML by replacing the ERM formulation with tilted losses. Following the setup in Collins et al. [57], we evaluate TERM on the popular sin wave regression problem. For a fair comparison, we perform task-level tilting for TERM, and operates on task-level reweighting for TR-MAML. From Table 7.6, we see that TERM with $t = 2$ not only decreases the standard deviation of test errors, but also achieves lower mean errors than MAML. As the number of tasks is large (5,000), solving the min-max variant (TR-MAML) is challenging, and results in slightly worse performance than TERM.

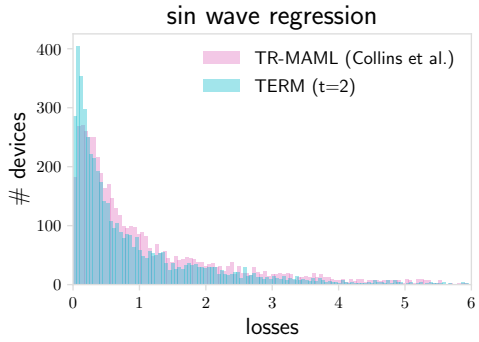


Figure 7.13: Loss distribution of TERM compared with the TR-MAML baseline.

Table 7.6: TERM ($t = 2$) results in fairer and lower test errors across meta-test tasks after adaptation compared with MAML [92]. TERM also outperforms a recently proposed min-max task-robust MAML method (TR-MAML) [57].

methods	mean	std	max	worst 10%
MAML	1.23	1.63	19.1	5.16
TR-MAML	1.25	1.51	14.31	4.85
TERM ($t = 2$)	**1.14**	**1.33**	**13.59**	**4.29**

7.7.2.4 Handling Class Imbalance

Next, we show that TERM can reduce the performance variance across classes with extremely imbalanced data when training deep neural networks. We compare TERM with several baselines which re-weight samples during training, including assigning weights inversely proportional to the class size (InverseRatio), focal loss [192], HardMine [201], and LearnReweight [238]. Following the setting of Ren et al. [238], the datasets are composed of imbalanced 4 and 9 digits from MNIST [165]. In Figure 7.14, we see that TERM obtains similar (or higher) final accuracy on the clean test data as the state-of-the-art methods. We note that compared with LearnReweight, which optimizes the model over an additional balanced validation set and requires three gradient calculations for each update, TERM neither requires such balanced validation data nor does it increase the per-iteration complexity.

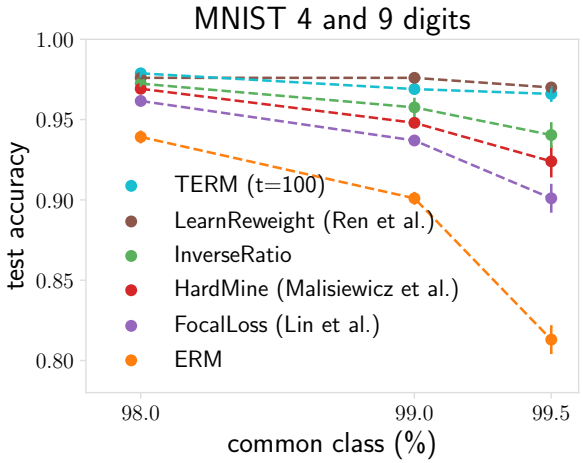


Figure 7.14: TERM (t=100) is competitive with state-of-the-art methods for classification with imbalanced classes.

7.7.2.5 Improving Generalization via Variance Reduction

A common alternative to ERM is to consider a distributionally robust objective, which optimizes for the worst-case training loss over a set of distributions, and has been shown to offer variance-reduction properties that benefit generalization [e.g., 51, 76, 77, 262]. While not directly developed for distributional robustness, TERM also enables variance reduction for positive values of t (Theorem 25), which can be used to strike a better bias-variance tradeoff for generalization. We compare TERM with several baselines including robustly regularized risk (RobustRegRisk) [77], linear SVM [238], Conditional Value-at-Risk (CVaR) [243, 268], LearnRewight [238], FocalLoss [192], and HRM [166] on the HIV-1 dataset [74, 245] originally investigated by Duchi and Namkoong [77]. We examine the accuracy on the rare class ($Y = 0$), the common class ($Y = 1$), and overall accuracy.

The mean and standard error of accuracies are reported in Table 7.7. RobustRegRisk and TERM offer similar performance improvements compared with other baselines, such as linear SVM, CVaR, LearnRewight, FocalLoss, and HRM. Note that here RobustRegRisk [77] and CVaR [243] can both be viewed as specific instances of the distributionally robust optimization framework, with different uncertainty sets. For larger t, TERM achieves similar accuracy in both classes, while RobustRegRisk does not show similar trends by sweeping its hyperparameters. It is common to adjust the decision threshold to boost the accuracy on the rare class. We do this for ERM and RobustRegRisk and optimize the threshold so that ERM_+ and RobustRegRisk_+ result in the same validation accuracy on the rare class as TERM ($t = 50$). TERM achieves similar performance to RobustRegRisk_+, without the need for an extra tuned hyperparameter.

Table 7.7: TERM ($t = 0.1$) is competitive with strong baselines in generalization. TERM ($t = 50$) outperforms ERM_+ (with decision threshold changed for providing fairness) and is competitive with RobustRegRisk_+ with no need for extra hyperparameter tuning.

objectives	**accuracy** ($Y = 0$)		**accuracy** ($Y = 1$)		**overall accuracy** (%)	
	train	test	train	test	train	test
ERM	0.841 (.005)	0.822 (.009)	0.971 (.000)	0.966 (.002)	0.944 (.000)	0.934 (.003)
Linear SVM	0.873 (.003)	**0.838** (.013)	0.965 (.000)	0.964 (.002)	0.951 (.001)	**0.937** (.004)
CVaR [243]	0.877 (.004)	**0.844** (.013)	0.972 (.000)	0.964 (.003)	0.952 (.001)	**0.937** (.003)
LearnReweight [238]	0.860 (.004)	**0.841** (.014)	0.960 (.002)	0.961 (.004)	0.940 (.001)	0.934 (.004)
FocalLoss [192]	0.871 (.003)	**0.834** (.013)	0.970 (.000)	0.966 (.003)	0.949 (.001)	**0.937** (.004)
HRM [166]	0.875 (.003)	**0.839** (.012)	0.972 (.000)	0.965 (.003)	0.952 (.001)	**0.937** (.003)
RobustRegRisk (Duchi et al., 2019)	0.875 (.003)	**0.844** (.010)	0.971 (.000)	0.966 (.003)	0.951 (.001)	**0.939** (.004)
TERM ($t = 0.1$)	0.864 (.003)	**0.840** (.011)	0.970 (.000)	0.964 (.003)	0.949 (.001)	**0.937** (.004)
ERM_+ (thresh = 0.26)	0.943 (.001)	0.916 (.008)	0.919 (.001)	0.917 (.003)	0.924 (.001)	0.917 (.002)
RobustRegRisk_+ (thresh=0.49)	0.943 (.000)	0.917 (.005)	0.928 (.001)	**0.928** (.002)	0.931 (.001)	**0.924** (.001)
TERM ($t = 50$)	0.942 (.001)	0.917 (.005)	0.926 (.001)	**0.925** (.002)	0.929 (.001)	**0.924** (.001)

7.7.3 Solving Compound Issues: Hierarchical Multi-Objective Tilting

Finally, in this section, we focus on settings where multiple issues, e.g., class imbalance and label noise, exist in the data simultaneously. We discuss two possible instances of hierarchical multi-objective TERM to tackle such problems. One can think of other variants in this hierarchical tilting space which could be useful depending on applications at hand.

7.7.3.1 Class Imbalance and Random Noise

We explore the HIV-1 dataset [245], as in Section 7.7.2. We report both overall accuracy and accuracy on the rare class in four scenarios: **(a) clean and 1:4**, the original dataset that is naturally slightly imbalanced with rare samples represented 1:4 with respect to the common class; **(b) clean and 1:20**, where we subsample to introduce a 1:20 imbalance ratio; **(c) noisy and 1:4**, which is the original dataset with labels associated with 30% of the samples randomly reshuffled; and **(d) noisy and 1:20**, where 30% of the labels of the 1:20 imbalanced dataset are reshuffled.

Table 7.8: Hierarchical TERM can address both class imbalance and noisy samples.

objectives	**test accuracy** (HIV-1)							
	clean data				30% noise			
	1:4		1:20		1:4		1:20	
	$Y=0$	overall	$Y=0$	overall	$Y=0$	overall	$Y=0$	overall
ERM	0.822 (.009)	0.934 (.003)	0.503 (.013)	0.888 (.006)	0.656 (.014)	0.911 (.006)	0.240 (.018)	0.831 (.011)
CVaR [243]	**0.844** (.013)	**0.937** (.003)	0.621 (.011)	0.906 (.005)	0.651 (.015)	0.909 (.006)	0.252 (.014)	0.834 (.010)
GCE [326]	0.822 (.009)	0.934 (.003)	0.503 (.013)	0.888 (.006)	0.732 (.021)	**0.925** (.005)	0.324 (.017)	0.849 (.008)
LearnReweight [238]	**0.841** (.014)	0.934 (.004)	0.800 (.022)	0.904 (.003)	0.721 (.034)	0.856 (.008)	0.532 (.054)	0.856 (.013)
RobustRegRisk (Duchi et al., 2019)	**0.844** (.010)	**0.939** (.004)	0.622 (.011)	0.906 (.005)	0.634 (.014)	0.907 (.006)	0.051 (.014)	0.792 (.012)
FocalLoss [192]	**0.834** (.013)	**0.937** (.004)	**0.806** (.020)	**0.918** (.003)	0.638 (.008)	0.908 (.005)	0.565 (.027)	**0.890** (.009)
HAR [45]	**0.842** (.011)	0.936 (.004)	0.817 (.013)	**0.926** (.004)	**0.870** (.010)	0.915 (.004)	**0.800** (.016)	0.867 (.012)
TERM_{sc}	**0.840** (.010)	**0.937** (.004)	**0.836** (.018)	**0.921** (.002)	**0.852** (.010)	**0.924** (.004)	**0.778** (.008)	**0.900** (.005)
TERM_{ca}	**0.844** (.014)	**0.938** (.004)	**0.834** (.021)	**0.918** (.003)	**0.846** (.015)	**0.933** (.003)	**0.806** (.020)	**0.901** (.010)

In Table 7.8, hierarchical TERM is applied at the sample level and class level (TERM_{sc}), where we use the sample-level tilt of $\tau=-2$ for noisy data. We use class-level tilt of $t=0.1$ for the 1:4 case and $t=50$ for the 1:20 case. We compare against baselines for robust classification and class imbalance (discussed previously in Sections 7.7.1 and 7.7.2), where we tune them for best performance (Appendix 7.12.2). Similar to the experiments in Section 7.7.1, we avoid using baselines that require clean validation data [e.g., 244]. We compare TERM with an additional baseline of HAR [45], a recent work addressing the issues of noisy and rare samples simultaneously with adaptive Lipschitz regularization. While different baselines (except HAR) perform well in their respective problem settings, TERM and HAR are far superior to all baselines when considering noisy samples and class imbalance simultaneously (rightmost column in Table 7.8). Finally, in the last row of Table 7.8, we simulate the noisy annotator setting of Section 7.7.1.3 assuming that the data is coming from 10 annotators, i.e., in the 30% noise case we have 7 hammers and 3

spammers. In this case, we apply hierarchical TERM at both class and annotator levels (TERM_{ca}), where we perform the higher level tilt at the annotator (group) level and the lower level tilt at the class level (with no sample-level tilting). We show that this approach can benefit noisy/imbalanced data even further (far right, Table 7.8), while suffering only a small performance drop on the clean and noiseless data (far left, Table 7.8).

7.7.3.2 Class Imbalance and Adversarial Noise

We evaluate hierarchical tilting on a more difficult task involving more adversarial noise with deep neural network models. We take the setup studied in Cao et al. [45]. The noise is created by exchanging labels of 40% samples which come from similar classes ('cat' and 'dog', 'vehicle' and 'automobile') in the CIFAR10 dataset. To simulate class imbalance, only 10% of the training data from these four noisy classes are subsampled. For TERM, we apply group-level positive tilting by linearly scaling t from 0 to 3, and perform sample-level negative tilting within each class with τ scaling from 0 to -2. Table 7.9 reports the results of hierarchical TERM (TERM_{sc}) compared with HAR [45] and other baselines. We see that TERM underperforms HAR, and outperforms all other approaches. Note that HAR is a more complicated method which requires to perform end-to-end training for two times with higher per-iteration complexity (involving second-order information), while TERM is a simple method and enjoys the same training time as that of ERM on this problem.

Table 7.9: TERM outperforms most baselines addressing the co-existence of noisy samples and class imbalance by a large margin, and is worse than a more complicated method HAR.

objectives	**test accuracy** (`CIFAR10`, ResNet32)	
	noisy, rare class	clean, common class
ERM	0.529 (.012)	**0.944** (.001)
GCE [326]	0.482 (.006)	0.916 (.003)
MentorNet [130]	0.541 (.010)	0.903 (.005)
MW-Net [259]	0.554 (.011)	0.917 (.005)
HAR [45]	**0.635** (.008)	**0.943** (.002)
TERM_{sc}	0.585 (.014)	0.913 (.003)

7.8 Discussion and Conclusion

In this chapter, we have explored the use of exponential tilting in risk minimization, examining tilted empirical risk minimization (TERM) as a flexible extension to the ERM framework. We rigorously established connections between TERM and related objectives including VaR, CVaR, and DRO. We explored, both theoretically and empirically, TERM's ability to handle various known issues with ERM, such as robustness to noise, class

imbalance, fairness, and generalization, as well as more complex issues like the simultaneous existence of class imbalance and noisy outliers. Despite the straightforward modification TERM makes to traditional ERM objectives, the framework consistently outperforms ERM and delivers competitive performance with state-of-the-art, problem-specific methods on a wide range of applications.

Our work highlights the effectiveness and versatility of tilted objectives in machine learning. As such, our framework (TERM) could be widely used for applications both positive and negative. However, our hope is that the TERM framework will allow machine learning practitioners to easily modify the ERM objective to handle practical concerns such as enforcing fairness amongst subgroups, mitigating the effect of outliers, and ensuring robust performance on new, unseen data. One potential downside of the TERM objective is that if the underlying dataset is *not* well-understood, incorrectly tuning t could have the unintended consequence of *magnifying* the impact of biased/corrupted data in comparison to traditional ERM. Indeed, critical to the success of such a framework is understanding the implications of the modified objective, both theoretically and empirically. The goal of this work is therefore to explore these implications so that it is clear when such a modified objective would be appropriate.

In terms of the use-cases explored with the TERM framework, we relied on benchmark datasets that have been commonly explored in prior work [e.g., 247, 281, 306, 314]. However, we note that some of these common benchmarks, such as cal-housing [226] and Credit [308], contain potentially sensitive information. While the goal of our experiments was to showcase that the TERM framework could be useful in learning fair representations that suppress membership bias and hence promote fairer performance, developing an understanding for—and removing—such membership biases requires a more comprehensive treatment of the problem that is outside the scope of this work.

In the future, in addition to generalization bounds of TERM, it would be interesting to further explore applications of tilted losses in machine learning. We note that since the early TERM work [183] was made public, there are several subsequent works applying (variants of) TERM to handle other real-world ML applications [279, 331], or exploring risk bounds on differential private TERM [197], which suggest rich implications and wide applicability of TERM, beyond what is studied in this work.

Here, we provide full statements and proofs of the analyses presented in Section 7.2-Section 7.4 (Appendix 7.9 and 7.10); details and convergence proof on the methods we propose for solving TERM (Appendix 7.11), and complete empirical results and details of our empirical setup (Appendix 7.12).

7.9 Properties and Interpretations (Proofs and Additional Results)

In this section, we provide the proofs of the main results in the work, along with additional results on the properties of TERM objective, its solution, as well as the corresponding solvers.

7.9.1 Proofs of Basic Properties of the TERM Objective

We first provide proofs for the basic properties of the TERM objective.

Proof of Lemma 22. The conclusion follows by noting that for any $\theta_1, \theta_2 \in \Theta$,

$$\left|\widetilde{R}(t;\theta_1) - \widetilde{R}(t;\theta_2)\right| = \left|\frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta_1)}\right) - \frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta_2)}\right)\right| \tag{7.87}$$

$$= \left|\frac{1}{t}\log\left(\frac{\sum_{i\in[N]} e^{tf(x_i;\theta_1)}}{\sum_{i\in[N]} e^{tf(x_i;\theta_2)}}\right)\right| \tag{7.88}$$

$$\leqslant \left|\frac{1}{t}\log\left(e^{tL\|\theta_1-\theta_2\|_2}\frac{\sum_{i\in[N]} e^{tf(x_i;\theta_2)}}{\sum_{i\in[N]} e^{tf(x_i;\theta_2)}}\right)\right| \tag{7.89}$$

$$= L\|\theta_1 - \theta_2\|_2. \tag{7.90}$$

□

Proof of Lemma 23. Recall that

$$\nabla_\theta \widetilde{R}(t;\theta) = \frac{\sum_{i\in[N]} \nabla_\theta f(x_i;\theta) e^{tf(x_i;\theta)}}{\sum_{i\in[N]} e^{tf(x_i;\theta)}} \tag{7.91}$$

$$= \frac{1}{N}\sum_{i\in[N]} \nabla_\theta f(x_i;\theta) e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}. \tag{7.92}$$

The proof of the first part is completed by differentiating again with respect to θ, followed by algebraic manipulation. To prove the second part, notice that the term in (7.13) is positive semi-definite, whereas the term in (7.14) is positive definite and lower bounded by $\beta_{\min}\mathbf{I}$ (see Assumption 9, Eq. (7.6)). □

Proof of Lemma 24. Let us first provide a proof for $t \in \mathbb{R}^{-}$. Invoking Lemma 23 and Weyl's inequality [298], we have

$$\begin{aligned}
&\lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right) \\
&\quad\leqslant \lambda_{\max}\left(\frac{t}{N}\sum_{i\in[N]}(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))^\top e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}\right) && (7.93)\\
&\quad+\lambda_{\max}\left(\frac{1}{N}\sum_{i\in[N]}\nabla^2_{\theta\theta^\top}f(x_i;\theta)e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}\right) && (7.94)\\
&\quad\leqslant \beta_{\max}, && (7.95)
\end{aligned}$$

where we have used the fact that the term in (7.13) is negative semi-definite for $t<0$, and that the term in (7.14) is positive definite for all t with smoothness bounded by $\beta_{\max}$ (which would hold from smoothness of $f(x_i;\theta)$; see Assumption 9, Eq. (7.6)).

For $t \in \mathbb{R}^{>0}$, following Lemma 23 and Weyl's inequality [298], we have

$$\begin{aligned}
&\left(\frac{1}{t}\right)\lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right) \\
&\quad\leqslant \lambda_{\max}\left(\frac{1}{N}\sum_{i\in[N]}(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))^\top e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}\right) && (7.96)\\
&\quad+\left(\frac{1}{t}\right)\lambda_{\max}\left(\frac{1}{N}\sum_{i\in[N]}\nabla^2_{\theta\theta^\top}f(x_i;\theta)e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}\right). && (7.97)
\end{aligned}$$

Due to Weyl's inequality, the smoothness of $f(x_i;\theta)$, and the fact that $\frac{1}{N}\sum_{i\in[N]}e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}=1$, $\sum_{i\in[N]}\nabla^2_{\theta\theta^\top}f(x_i;\theta)e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}$ is bounded. Consequently,

$$\lim_{t\to+\infty}\left(\frac{1}{t}\right)\lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right)<+\infty. \tag{7.98}$$

On the other hand, following Weyl's inequality [298],

$$\begin{aligned}
&\lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right) \\
&\quad\geqslant t\lambda_{\max}\left(\frac{1}{N}\sum_{i\in[N]}(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))(\nabla_\theta f(x_i;\theta)-\nabla_\theta\widetilde{R}(t;\theta))^\top e^{t(f(x_i;\theta)-\widetilde{R}(t;\theta))}\right), && (7.99)
\end{aligned}$$

and hence,

$$\lim_{t\to+\infty}\left(\frac{1}{t}\right)\lambda_{\max}\left(\nabla^2_{\theta\theta^\top}\widetilde{R}(t;\theta)\right)>0, \tag{7.100}$$

where we have used the fact that no solution θ exists that would make all f_i's vanish (Assumption 9). □

Under the strict saddle property (Assumption 11), it is known that gradient-based methods would converge to a local minimum [97], i.e., $\breve{\theta}(t)$ would be obtained using gradient descent (GD). The rate of convergence of GD scales linearly with the smoothness parameter of the optimization landscape, which is characterized by Lemma 24.

Proof of Lemma 25. For $t\to 0$,

$$\begin{aligned}\lim_{t\to0}\widetilde{R}(t;\theta)&=\lim_{t\to0}\frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]}e^{tf(x_i;\theta)}\right)\\&=\lim_{t\to0}\frac{\sum_{i\in[N]}f(x_i;\theta)e^{tf(x_i;\theta)}}{\sum_{i\in[N]}e^{tf(x_i;\theta)}} &(7.101)\\&=\frac{1}{N}\sum_{i\in[N]}f(x_i;\theta), &(7.102)\end{aligned}$$

where (7.101) is due to L'Hôpital's rule applied to t as the denominator and $\log\left(\frac{1}{N}\sum_{i\in[N]}e^{tf(x_i;\theta)}\right)$ as the numerator.

For $t\to-\infty$, we proceed as follows:

$$\begin{aligned}\lim_{t\to-\infty}\widetilde{R}(t;\theta)&=\lim_{t\to-\infty}\frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]}e^{tf(x_i;\theta)}\right)\\&\leqslant\lim_{t\to-\infty}\frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]}e^{t\min_{j\in[N]}f(x_j;\theta)}\right) &(7.103)\\&=\min_{i\in[N]}f(x_i;\theta). &(7.104)\end{aligned}$$

On the other hand,

$$\begin{aligned}\lim_{t\to-\infty}\widetilde{R}(t;\theta)&=\lim_{t\to-\infty}\frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]}e^{tf(x_i;\theta)}\right)\\&\geqslant\lim_{t\to-\infty}\frac{1}{t}\log\left(\frac{1}{N}e^{t\min_{j\in[N]}f(x_j;\theta)}\right) &(7.105)\\&=\min_{i\in[N]}f(x_i;\theta)-\lim_{t\to-\infty}\left\{\frac{1}{t}\log N\right\} &(7.106)\end{aligned}$$

$$= \min_{i \in [N]} f(x_i; \theta). \tag{7.107}$$

Hence, the proof follows by putting together (7.104) and (7.107).

The proof proceeds similarly to $t \to -\infty$ for $t \to +\infty$ and is omitted for brevity. □

7.9.2 General Properties of the Objective for GLMs

In this section, even if not explicitly stated, all results are derived under Assumption 10 with a generalized linear model and loss function of the form (7.7), effectively assuming that the loss function is the negative log-likelihood of an exponential family [286].

Definition 16 (Empirical cumulant generating function)**.** *Let*

$$\widetilde{\Lambda}(t; \theta) := t\widetilde{R}(t; \theta). \tag{7.108}$$

Definition 17 (Empirical log-partition function [287])**.** *Let* $\Gamma(t; \theta)$ *be*

$$\Gamma(t; \theta) := \log \left(\frac{1}{N} \sum_{i \in [N]} e^{-t\theta^\top T(x_i)} \right). \tag{7.109}$$

Thus, we have

$$\widetilde{R}(t; \theta) = A(\theta) + \frac{1}{t} \log \left(\frac{1}{N} \sum_{i \in [N]} e^{-t\theta^\top T(x_i)} \right) = A(\theta) + \frac{1}{t}\Gamma(t; \theta). \tag{7.110}$$

Definition 18 (Tilted empirical mean and empirical variance of the sufficient statistic)**.** *Let* $\mathcal{M}$ *and* $\mathcal{V}$ *denote the mean and the variance of the sufficient statistic, and be given by*

$$\mathcal{M}(t; \theta) := \frac{1}{N} \sum_{i \in [N]} T(x_i) e^{-t\theta^\top T(x_i) - \Gamma(t;\theta)}, \tag{7.111}$$

$$\mathcal{V}(t; \theta) := \frac{1}{N} \sum_{i \in [N]} (T(x_i) - \mathcal{M}(t; \theta))(T(x_i) - \mathcal{M}(t; \theta))^\top e^{-t\theta^\top T(x_i) - \Gamma(t;\theta)}. \tag{7.112}$$

We notice that $\mathcal{M}(t; \theta)$ and $\mathcal{V}(t; \theta)$ defined here are equivalent to tilted empirical mean/variance in the main text (Eq. (7.29) and Eq. (7.31)) over sufficient statistic, i.e.,

$$\mathcal{M}(t; \theta) = \sum_{i \in [N]} w_i(t; \theta) T(x_i), \tag{7.113}$$

$$\mathcal{V}(t; \theta) = \sum_{i \in [N]} w_i(t; \theta)(T(x_i) - \mathcal{M}(t; \theta))(T(x_i) - \mathcal{M}(t; \theta))^\top. \tag{7.114}$$

Similarly, as a special case of t-tilted empirical mean/variance (Eq. (7.30) and Eq. (7.32)), t-tilted empirical mean/variance over sufficient statistic are defined as

$$\mathcal{M}_t := \mathcal{M}(t; \breve{\theta}(t)), \tag{7.115}$$

$$\mathcal{V}_t := \mathcal{V}(t; \breve{\theta}(t)). \tag{7.116}$$

The quantities $\mathcal{M}(t;\theta), \mathcal{V}(t;\theta), \mathcal{M}_t$, and $\mathcal{V}_t$ will be used for proving general properties of TERM solutions in this section.

Lemma 33. *For all $t \in \mathbb{R}$, we have $\mathcal{V}(t;\theta) \succeq 0$.*

Next we state a few key relationships that we will use in our characterizations. The proofs are straightforward and omitted for brevity.

Lemma 34 (Partial derivatives of Γ)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\Gamma(t;\theta) = -\theta^\top \mathcal{M}(t;\theta), \tag{7.117}$$

$$\nabla_\theta \Gamma(t;\theta) = -t\mathcal{M}(t;\theta). \tag{7.118}$$

Lemma 35 (Partial derivatives of $\mathcal{M}$)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\mathcal{M}(t;\theta) = -\mathcal{V}(t;\theta)\theta, \tag{7.119}$$

$$\nabla_\theta \mathcal{M}(t;\theta) = -t\mathcal{V}(t;\theta). \tag{7.120}$$

The next few lemmas characterize the partial derivatives of the cumulant generating function.

Lemma 36. *(Derivative of $\widetilde{\Lambda}$ with t) For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\widetilde{\Lambda}(t;\theta) = A(\theta) - \theta^\top \mathcal{M}(t;\theta). \tag{7.121}$$

Proof. The proof is carried out by

$$\frac{\partial}{\partial t}\widetilde{\Lambda}(t;\theta) = A(\theta) - \theta^\top \sum_{i\in[N]} T(x_i) e^{-t\theta^\top T(x_i) - \Gamma(t;\theta)} = A(\theta) - \theta^\top \mathcal{M}(t;\theta). \tag{7.122}$$

□

Lemma 37 (Second derivative of $\widetilde{\Lambda}$ with t)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\frac{\partial^2}{\partial t^2}\widetilde{\Lambda}(t;\theta) = \theta^\top \mathcal{V}(t;\theta)\theta. \tag{7.123}$$

Lemma 38 (Gradient of $\widetilde{\Lambda}$ with θ)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\nabla_\theta \widetilde{\Lambda}(t;\theta) = t\nabla_\theta A(\theta) - t\mathcal{M}(t;\theta). \tag{7.124}$$

Lemma 39 (Hessian of $\widetilde{\Lambda}$ with θ)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\nabla^2_{\theta\theta^\top} \widetilde{\Lambda}(t;\theta) = t\nabla^2_{\theta\theta^\top} A(\theta) + t^2 \mathcal{V}(t;\theta). \tag{7.125}$$

Lemma 40 (Gradient of $\widetilde{\Lambda}$ with respect to t and θ)**.** *For all $t \in \mathbb{R}$ and all $\theta \in \Theta$,*

$$\frac{\partial}{\partial t}\nabla_\theta \widetilde{\Lambda}(t;\theta) = \nabla_\theta A(\theta) - \mathcal{M}(t;\theta) + t\mathcal{V}(t;\theta)\theta. \tag{7.126}$$

Proof of Theorem 23. Following (7.110),

$$\frac{\partial}{\partial t}\widetilde{R}(t;\theta) = \frac{\partial}{\partial t}\left\{\frac{1}{t}\Gamma(t;\theta)\right\} \tag{7.127}$$

$$= -\frac{1}{t^2}\Gamma(t;\theta) - \frac{1}{t}\theta^\top \mathcal{M}(t;\theta), \tag{7.128}$$

$$=: g(t;\theta), \tag{7.129}$$

where (7.128) follows from Lemma 34, and (7.129) defines $g(t;\theta)$.

Let $g(0;\theta) := \lim_{t\to 0} g(t;\theta)$ Notice that

$$g(0;\theta) = \lim_{t\to 0}\left\{-\frac{1}{t^2}\Gamma(t;\theta) - \frac{1}{t}\theta^\top \mathcal{M}(t;\theta)\right\} \tag{7.130}$$

$$= -\lim_{t\to 0}\left\{\frac{\frac{1}{t}\Gamma(t;\theta) + \theta^\top \mathcal{M}(t;\theta)}{t}\right\} \tag{7.131}$$

$$= \theta^\top \mathcal{V}(0;\theta)\theta, \tag{7.132}$$

where (7.132) is due to L'H$\hat{0}$pital's rule and Lemma 37. Now consider

$$\frac{\partial}{\partial t}\left\{t^2 g(t;\theta)\right\} = \frac{\partial}{\partial t}\left\{-\Gamma(t;\theta) - t\theta^\top \mathcal{M}(t;\theta)\right\} \tag{7.133}$$

$$= \theta^\top \mathcal{M}(t;\theta) \tag{7.134}$$

$$- \theta^\top \mathcal{M}(t;\theta) + t\theta^\top \mathcal{V}(t;\theta)\theta \tag{7.135}$$

$$= t\theta^\top \mathcal{V}(t;\theta)\theta, \tag{7.136}$$

where $g(t;\theta) = \frac{\partial}{\partial t}\widetilde{R}(t;\theta)$, (7.134) follows from Lemma 34, (7.135) follows from the chain rule and Lemma 35. Hence, $t^2 g(t;\theta)$ is an increasing function of t for $t \in \mathbb{R}^{>0}$, and a decreasing function of t for $t \in \mathbb{R}^{-}$, taking its minimum at $t = 0$. Hence, $t^2 g(t;\theta) \geqslant 0$ for all $t \in \mathbb{R}$. This implies that $g(t;\theta) \geqslant 0$ for all $t \in \mathbb{R}$, which in conjunction with (7.129) implies the statement of the theorem.

7.9.3 General Properties of TERM Solutions for GLMs

Next, we characterize some of the general properties of the solutions of TERM objectives. Note that these properties are established under Assumptions 10 and 11.

Lemma 41. *For all $t \in \mathbb{R}$,*

$$\nabla_\theta \widetilde{\Lambda}(t;\breve{\theta}(t)) = 0. \tag{7.137}$$

Proof. The proof follows from definition and the assumption that Θ is an open set. □

Lemma 42. *For all $t \in \mathbb{R}$,*

$$\nabla_\theta A(\breve{\theta}(t)) = \mathcal{M}(t;\breve{\theta}(t)). \tag{7.138}$$

Proof. The proof is completed by noting Lemma 41 and Lemma 38. □

Lemma 43 (Derivative of the solution with respect to tilt)**.** *Under Assumption 11, for all* $t \in \mathbb{R}$,

$$\frac{\partial}{\partial t}\breve{\theta}(t) = -\left(\nabla^2_{\theta\theta^\top} A(\breve{\theta}(t)) + t\mathcal{V}(t;\breve{\theta}(t))\right)^{-1} \mathcal{V}(t;\breve{\theta}(t))\breve{\theta}(t), \tag{7.139}$$

where

$$\nabla^2_{\theta\theta^\top} A(\breve{\theta}(t)) + t\mathcal{V}(t;\breve{\theta}(t)) \succ 0 \tag{7.140}$$

is a symmetric positive definite matrix.

Proof. By noting Lemma 41, and further differentiating with respect to t, we have

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}\nabla_\theta \widetilde{\Lambda}(t;\breve{\theta}(t)) && (7.141)\\ &= \frac{\partial}{\partial \tau}\nabla_\theta \widetilde{\Lambda}(\tau;\breve{\theta}(t))\bigg|_{\tau=t} + \nabla^2_{\theta\theta^\top}\widetilde{\Lambda}(t;\breve{\theta}(t))\left(\frac{\partial}{\partial t}\breve{\theta}(t)\right) && (7.142)\\ &= t\mathcal{V}(t;\breve{\theta}(t))\breve{\theta}(t) + \left(t\nabla^2_{\theta\theta^\top} A(\theta) + t^2\mathcal{V}(t;\theta)\right)\left(\frac{\partial}{\partial t}\breve{\theta}(t)\right), && (7.143) \end{aligned}$$

where (7.142) follows from the chain rule, (7.143) follows from Lemmas 40 and 42 and 39. The proof is completed by noting that $\nabla^2_{\theta\theta^\top}\widetilde{\Lambda}(t;\breve{\theta}(t))$ is symmetric positive definite for all $t \in \mathbb{R}$ under Assumption 11. □

Finally, we state an auxiliary lemma that will be used in the proof of the main theorem.

Lemma 44. *For all* $t, \tau \in \mathbb{R}$ *and all* $\theta \in \Theta$,

$$\mathcal{M}(\tau;\theta) - \mathcal{M}(t;\theta) = -\left(\int_t^\tau \mathcal{V}(\nu;\theta)d\nu\right)\theta. \tag{7.144}$$

Proof. The proof is completed by noting that

$$\mathcal{M}(\tau;\theta) - \mathcal{M}(t;\theta) = \int_t^\tau \frac{\partial}{\partial \nu}\mathcal{M}(\nu;\theta)d\nu = -\left(\int_t^\tau \mathcal{V}(\nu;\theta)d\nu\right)\theta. \tag{7.145}$$

□

Proof of Theorem 24. Notice that for all θ, and all $\epsilon \in \mathbb{R}^{>0}$,

$$\begin{aligned} \widetilde{R}(t+\epsilon;\theta) &\geqslant \widetilde{R}(t;\theta) && (7.146)\\ &\geqslant \widetilde{R}(t;\breve{\theta}(t)), && (7.147) \end{aligned}$$

where (7.146) follows from Theorem 23 and (7.147) follows from the definition of $\breve{\theta}(t)$. Hence,

$$\widetilde{R}(t+\epsilon;\breve{\theta}(t+\epsilon)) = \min_{\theta\in B(\breve{\theta}(t),r)} \widetilde{R}(t+\epsilon;\theta) \geqslant \widetilde{R}(t;\breve{\theta}(t)), \tag{7.148}$$

which completes the proof. □

Proof of Theorem 25. Recall that $f(x_i;\theta) = A(\theta) - \theta^\top T(x_i)$. Thus,

$$\widehat{E}_t(\mathbf{f}(\theta)) = \sum_{i\in[N]} w_i(t;\breve{\theta}(t)) f(x_i;\theta) = A(\theta) - \theta^\top \sum_{i\in[N]} w_i(t;\breve{\theta}(t)) T(x_i) = A(\theta) - \theta^\top \mathcal{M}_t, \tag{7.149}$$

where $\mathcal{M}_t$ is defined in (7.115). Consequently,

$$\widehat{\mathbf{Var}}_t(\mathbf{f}(\theta)) = \widehat{E}_t\left(f(x_i;\theta) - \widehat{E}_t(\mathbf{f}(\theta)\right)^2 \tag{7.150}$$

$$= \widehat{E}_t\left(\theta^\top T(x_i) - \theta^\top \mathcal{M}_t\right)^2 \tag{7.151}$$

$$= \theta^\top \widehat{E}_t\left((T(x_i) - \mathcal{M}_t)(T(x_i) - \mathcal{M}_t)^\top\right)\theta \tag{7.152}$$

$$= \theta^\top \mathcal{V}_t \theta, \tag{7.153}$$

where $\mathcal{V}_t$ is defined in (7.116). Hence,

$$\frac{\partial}{\partial\tau}\left\{\widehat{\mathbf{Var}}_t(\mathbf{f}(\breve{\theta}(\tau)))\right\} = \left(\frac{\partial}{\partial\tau}\breve{\theta}(\tau)\right)^\top \nabla_\theta \left\{\widehat{\mathbf{Var}}_t(\mathbf{f}(\breve{\theta}(\tau)))\right\} \tag{7.154}$$

$$= 2\left(\frac{\partial}{\partial\tau}\breve{\theta}(\tau)\right)^\top \mathcal{V}_t \breve{\theta}(\tau) \tag{7.155}$$

$$= -2\breve{\theta}^\top(\tau)\mathcal{V}(\tau;\breve{\theta}(\tau))\left(\nabla^2_{\theta\theta}A(\breve{\theta}(\tau)) + \tau\mathcal{V}(\tau;\breve{\theta}(\tau))\right)^{-1}\mathcal{V}_t\breve{\theta}(\tau), \tag{7.156}$$

and in turn

$$\frac{\partial}{\partial\tau}\left\{\widehat{\mathbf{Var}}_t(\mathbf{f}(\breve{\theta}(\tau)))\right\}\Big|_{\tau=t} \leqslant 0, \tag{7.157}$$

where we have used the fact that $\mathcal{V}_\tau\left(\nabla^2_{\theta\theta}A(\breve{\theta}(\tau)) + \tau\mathcal{V}_\tau\right)^{-1}\mathcal{V}_\tau$ is a symmetric positive semidefinite matrix (due to Lemma 33), hence completing the proof. □

Proof of Theorem 26. Notice that

$$H\left(\mathbf{w}(t;\theta)\right) = -\sum_{i\in[N]} w_i(t;\theta)\log w_i(t;\theta) \tag{7.158}$$

$$= -\frac{1}{N}\sum_{i\in[N]}(tf(x_i;\theta) - \widetilde{\Lambda}(t;\theta))e^{tf(x_i;\theta)-\widetilde{\Lambda}(t;\theta)} \tag{7.159}$$

$$= \widetilde{\Lambda}(t;\theta) - t\frac{1}{N}\sum_{i\in[N]} f(x_i;\theta)e^{tf(x_i;\theta)-\widetilde{\Lambda}(t;\theta)} \tag{7.160}$$

$$= \widetilde{\Lambda}(t;\theta) - tA(\theta) + t\theta^\top\mathcal{M}(t;\theta). \tag{7.161}$$

Thus,

$$\nabla_\theta H\left(\mathbf{w}(t;\theta)\right) = \nabla_\theta\left(\widetilde{\Lambda}(t;\theta) - tA(\theta) + t\theta^\top\mathcal{M}(t;\theta)\right) \tag{7.162}$$

$$= t\nabla_\theta A(\theta) - t\mathcal{M}(t;\theta) - t\nabla_\theta A(\theta) + t\mathcal{M}(t;\theta) - t^2\mathcal{V}(t;\theta)\theta \tag{7.163}$$

$$= -t^2\mathcal{V}(t;\theta)\theta. \tag{7.164}$$

Hence,

$$\frac{\partial}{\partial\tau}H\left(\mathbf{w}(t;\breve{\theta}(\tau))\right) = \left(\frac{\partial}{\partial\tau}\breve{\theta}(\tau)\right)^\top \nabla_\theta H\left(\mathbf{w}(t;\breve{\theta}(\tau))\right) \tag{7.165}$$

$$= \left(\frac{\partial}{\partial\tau}\breve{\theta}(\tau)\right)^\top \nabla_\theta\left(\widetilde{\Lambda}(t;\theta) - tA(\theta) + t\theta^\top\mathcal{M}(t;\theta)\right) \tag{7.166}$$

$$= t^2\breve{\theta}^\top(\tau)\mathcal{V}(\tau;\breve{\theta}(\tau))\left(\nabla^2_{\theta\theta}A(\breve{\theta}(\tau)) + \tau\mathcal{V}(\tau;\breve{\theta}(\tau))\right)^{-1}\mathcal{V}(t;\breve{\theta}(\tau))\breve{\theta}(\tau) \tag{7.167}$$

and

$$\frac{\partial}{\partial\tau}H\left(\mathbf{w}(t;\breve{\theta}(\tau))\right)\Big|_{t=\tau} \geqslant 0, \tag{7.168}$$

completing the proof. □

There are different ways to define performance uniformity. In Theorem 35, we further prove that the tilted cosine similarity between the scaled loss vector and the all-ones vector increases as t decreases by a small amount, which shows that larger t promotes a more *uniform* performance across all losses and can have implications for fairness defined as representation disparity [113] (Section 7.7.2).

Definition 19 (t-tilted cosine similarity)**.** *For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$, let cosine similarity be defined as*

$$s(\mathbf{u}, \mathbf{v}) := \frac{\mathbf{u}^\top\mathbf{v}}{\|\mathbf{u}\|_2\|\mathbf{v}\|_2}. \tag{7.169}$$

For a weight vector $\mathbf{w}$, let the weighted cosine similarity be defined as

$$s_{\mathbf{w}}(\mathbf{u}, \mathbf{v}) := s\left(\sqrt{\mathbf{W}}\mathbf{u}, \sqrt{\mathbf{W}}\mathbf{v}\right), \tag{7.170}$$

where $\mathbf{W} := diag(\mathbf{w})$. In particular, we call $s_{\mathbf{w}(t;\breve{\theta}(t))}(\cdot,\cdot)$ the t-tilted cosine similarity.

Theorem 35 (t-tilted cosine similarity of the scaled loss vector and the all-ones vector increases with t)**.** *Let*

$$\mathbf{f}^+(\theta) := \left\{f(x_i;\theta) - \widetilde{F}(-\infty)\right\}_{i\in[N]}, \tag{7.171}$$

where $\widetilde{F}(-\infty)$ is defined in Eq. (7.10)*, and let $\mathbf{1}_N$ denote the all-one N-vector. Then, under Assumption 10 and Assumption 11, for any $t \in \mathbb{R}$,*

$$\frac{\partial}{\partial t}\left\{s_{\mathbf{w}(\tau,\breve{\theta}(\tau))}\left(\mathbf{f}^+(\breve{\theta}(t)), \mathbf{1}_N\right)\right\}\Big|_{\tau=t} > 0, \tag{7.172}$$

where $\mathbf{w}(t;\breve{\theta}(t))$ is the tilted weight vector defined in Eq. (7.26)*.*

Proof. Notice that

$$s_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{f}^+(\theta), \mathbf{1}_N) = \frac{\widehat{E}_t f(x_i;\theta) - \widetilde{F}(-\infty)}{\sqrt{\widehat{E}_t (f(x_i;\theta) - \widetilde{F}(-\infty))^2}}. \tag{7.173}$$

Hence,

$$\widehat{E}_t f(x_i;\theta) - \widetilde{F}(-\infty) = A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty), \tag{7.174}$$

$$\widehat{E}_t (f(x_i;\theta) - \widetilde{F}(-\infty))^2 = (A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \theta^\top \mathcal{V}_t \theta, \tag{7.175}$$

where $\mathcal{M}_t$ and $\mathcal{V}_t$ are defined in (7.115) and (7.116), respectively. Notice that

$$\nabla_\theta \left\{ s^2_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{f}^+(\theta), \mathbf{1}_N) \right\} \tag{7.176}$$

$$= \nabla_\theta \left\{ \frac{\left(\widehat{E}_t f(x_i;\theta) - \widetilde{F}(-\infty)\right)^2}{\widehat{E}_t (f(x_i;\theta) - \widetilde{F}(-\infty))^2} \right\} \tag{7.177}$$

$$= \nabla_\theta \left\{ \frac{(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2}{(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \theta^\top \mathcal{V}_t \theta} \right\} \tag{7.178}$$

$$= \frac{2(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))(\nabla_\theta A(\theta) - \mathcal{M}_t)\theta^\top \mathcal{V}_t \theta - 2(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 \mathcal{V}_t \theta}{\left((A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \theta^\top \mathcal{V}_t \theta\right)^2} \tag{7.179}$$

$$= \frac{2(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty)) \left(\theta^\top (\nabla_\theta A(\theta) - \mathcal{M}_t) - A(\theta) + \theta^\top \mathcal{M}_t + \widetilde{F}(-\infty)\right) \mathcal{V}_t \theta}{\left((A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \theta^\top \mathcal{V}_t \theta\right)^2} \tag{7.180}$$

$$= \frac{2(A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty)) \left(\theta^\top \nabla_\theta A(\theta) - A(\theta) + \widetilde{F}(-\infty)\right) \mathcal{V}_t \theta}{\left((A(\theta) - \theta^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \theta^\top \mathcal{V}_t \theta\right)^2}. \tag{7.181}$$

Hence,

$$\frac{\partial}{\partial \tau} \left\{ s^2_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{f}^+(\breve{\theta}(\tau)), \mathbf{1}_N) \right\} \tag{7.182}$$

$$= \left(\frac{\partial}{\partial \tau} \breve{\theta}(\tau)\right)^\top \nabla_\theta \left\{ s^2_{\mathbf{w}(t;\breve{\theta}(t))}(\mathbf{f}^+(\breve{\theta}(\tau)), \mathbf{1}_N) \right\} \tag{7.183}$$

$$= -\breve{\theta}^\top(\tau) \mathcal{V}(\tau; \breve{\theta}(\tau)) \left(\nabla^2_{\theta\theta} A(\breve{\theta}(\tau)) + \tau \mathcal{V}(\tau; \breve{\theta}(\tau))\right)^{-1}$$
$$\times - \frac{2(A(\breve{\theta}(\tau)) - \breve{\theta}(\tau)^\top \mathcal{M}_t - \widetilde{F}(-\infty))(A(\breve{\theta}(\tau)) - \breve{\theta}(\tau)^\top \mathcal{M}(\tau; \breve{\theta}(\tau)) - \widetilde{F}(-\infty))}{\left((A(\breve{\theta}(\tau)) - \breve{\theta}(\tau)^\top \mathcal{M}_t - \widetilde{F}(-\infty))^2 + \breve{\theta}(\tau)^\top \mathcal{V}_t \theta\right)^2} \mathcal{V}_t \breve{\theta}(\tau). \tag{7.184}$$

Note that $2(A(\breve{\theta}(\tau)) - \breve{\theta}(\tau)^\top \mathcal{M}_t - \widetilde{F}(-\infty))(A(\breve{\theta}(\tau)) - \breve{\theta}(\tau)^\top \mathcal{M}(\tau; \breve{\theta}(\tau)) - \widetilde{F}(-\infty)) > 0$ by definition, and $\mathcal{V}_\tau \left(\nabla^2_{\theta\theta} A(\breve{\theta}(\tau)) + \tau \mathcal{V}_\tau\right)^{-1} \mathcal{V}_\tau$ is a symmetric positive semi-definite matrix. Therefore, The proof is completed following that the quantity in Eq. (7.184) is non-negative for $t = \tau$. □

7.10 Connections to Other Objectives (Proofs and Additional Results)

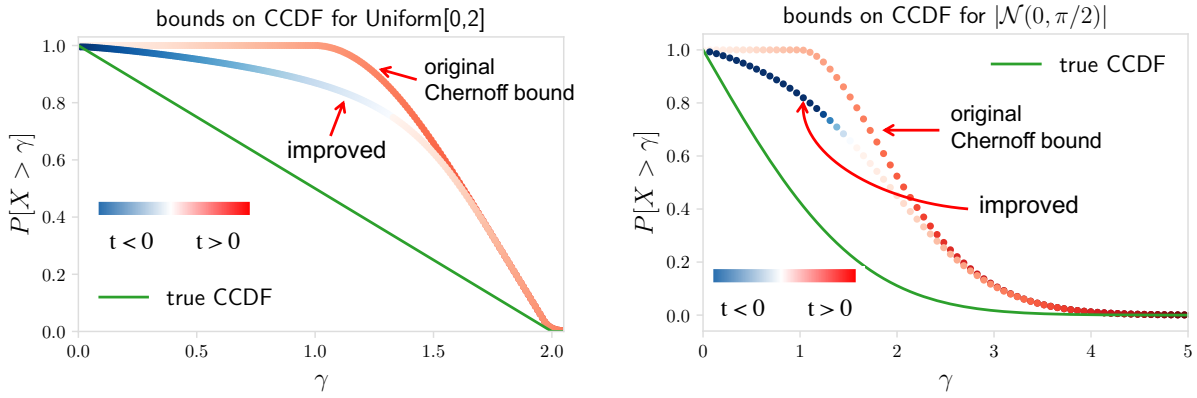


Figure 7.15: Comparing the new Chernoff bound on complementary CDF (CCDF) (i.e., $P[X \geqslant \gamma]$) proposed in Theorem 27 (denoted as 'improved') with the original Chernoff bound in two cases: $X \sim \text{Uniform}[0, 2]$ and $X \sim |\mathcal{N}(0, \pi/2)|$. We see that by sweeping t from all real numbers, our bound is significantly tighter than the generic Chernoff bound which optimizes over $t \in \mathbb{R}^+$, especially in the small deviations regime.

Lemma 45. *If $a < \widetilde{F}(-\infty)$ then $\widetilde{Q}^0(\gamma) = 1$. Further, if $\gamma > \widetilde{F}(+\infty)$ then $\widetilde{Q}^0(\gamma) = 0$, where $\widetilde{F}(\cdot)$ is defined in Definition 7.11, and is reproduced here:*

$$\widetilde{F}(-\infty) = \lim_{t \to -\infty} \widetilde{R}(t; \breve{\theta}(t)) = \min_{\theta} \min_{i \in [N]} f(x_i; \theta), \tag{7.185}$$

$$\widetilde{F}(+\infty) = \lim_{t \to +\infty} \widetilde{R}(t; \breve{\theta}(t)) = \min_{\theta} \max_{i \in [N]} f(x_i; \theta). \tag{7.186}$$

Next, we present our main result on the connection between tail distribution of losses and TERM, using Theorem 27.

Theorem 36. *For all $t \in \mathbb{R}$, and all θ, and all $\gamma \in (\widetilde{F}(-\infty), \widetilde{F}(+\infty))$,*[7]

$$\widetilde{Q}(\gamma; \theta) \leqslant \overline{Q}(\gamma; t, \theta) := \frac{e^{\widetilde{R}(t;\theta)t} - e^{\widetilde{F}(-\infty)t}}{e^{\gamma t} - e^{\widetilde{F}(-\infty)t}}. \tag{7.187}$$

Proof. The proof is a direct application of Theorem 27 to the non-negative random variable $(f(X; \theta) - \widetilde{F}(-\infty))$, where X is distributed according to the empirical distribution. □

[7] We define the RHS at $t = 0$ via continuous extension.

Recall that optimizing $\widetilde{\text{VaR}}$ is equivalent to optimizing $\widetilde{Q}$. Next we show how TERM is related to optimizing $\widetilde{Q}$. Recall that $\widetilde{Q}^0(\gamma)$ denotes the optimal value of $\widetilde{Q}(\gamma;\theta)$ optimized over θ. Let

$$\widetilde{Q}^1(\gamma) := \inf_{t\in\mathbb{R}} \left\{ \widetilde{Q}(\gamma; \breve{\theta}(t)) \right\}, \tag{7.188}$$

which denotes the value at risk optimized over the t-tilted solutions.

Theorem 37. *For all $\gamma \in (\widetilde{F}(-\infty), \widetilde{F}(+\infty))$, we have*

$$\widetilde{Q}^0(\gamma) \leqslant \widetilde{Q}^1(\gamma) \leqslant \widetilde{Q}^2(\gamma) \leqslant \widetilde{Q}^3(\gamma) = \inf_{t\in\mathbb{R}} \left\{ \overline{Q}(\gamma, t) \right\}, \tag{7.189}$$

where

$$\overline{Q}(\gamma, t) := \frac{e^{\widetilde{F}(t)t} - e^{\widetilde{F}(-\infty)t}}{e^{\gamma t} - e^{\widetilde{F}(-\infty)t}}, \tag{7.190}$$

$$\widetilde{t}^3(\gamma) := \arg\inf_{t\in\mathbb{R}} \left\{ \overline{Q}(\gamma, t) \right\}, \tag{7.191}$$

$$\widetilde{Q}^2(\gamma) := \widetilde{Q}(\gamma; \breve{\theta}(\widetilde{t}^3(\gamma))), \tag{7.192}$$

$$\widetilde{Q}^3(\gamma) := \overline{Q}(\gamma, \widetilde{t}^3(\gamma)). \tag{7.193}$$

Proof. The only non-trivial step is to show that $Q^2(\gamma) \leqslant Q^3(\gamma)$. Following Theorem 36,

$$Q^2(\gamma) = \widetilde{Q}(\gamma; \breve{\theta}(\widetilde{t}(\gamma)) \leqslant \inf_{t\in\mathbb{R}} \overline{Q}(\gamma; t, \breve{\theta}(t)) = Q^3(\gamma), \tag{7.194}$$

which completes the proof. □

Theorem 37 motivates us with the following approximation on the solutions of the minimizing the tail distribution of losses (Definition 14).

Approximation 1. *For all $\gamma \in (\widetilde{F}(-\infty), \widetilde{F}(+\infty))$,*

$$\widetilde{Q}(\gamma; \theta^0(\gamma)) = \widetilde{Q}^0(\gamma) \approx \widetilde{Q}^2(\gamma) = \widetilde{Q}(\gamma; \breve{\theta}(\widetilde{t}(\gamma))), \tag{7.195}$$

and hence, $\breve{\theta}(\widetilde{t}(\gamma))$ is an approximate solution to the tail probability optimization problem.

While we have not characterized how tight this approximation is for $\gamma \in (\widetilde{F}(-\infty), \widetilde{F}(+\infty))$, we believe that Approximation 1 provides a reasonable solution to the tail distribution optimization problem in general. This is evidenced empirically when the approximation is evaluated on the toy examples of Figure 7.1, and compared with the global solutions of the tail distribution optimization method, as shown in Figure 7.16. As can be seen, $\widetilde{Q}^0(\gamma) \approx \widetilde{Q}^2(\gamma)$ as suggested by Approximation 1. Also, we can see that while the bound in Theorem 37 ($\widetilde{Q}^3(\gamma)$) is not tight, the solution that is obtained from solving it ($\widetilde{Q}^2(\gamma)$) results in a good approximation to the tail distribution minimization ($\widetilde{Q}^0(\gamma)$).

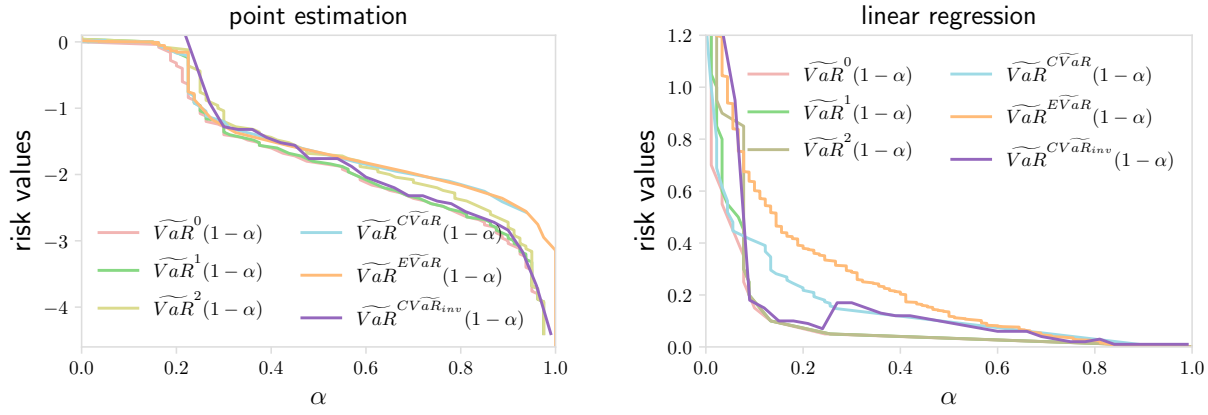


Figure 7.16: Comparing the solutions of different risks in terms of how well they solve VaR. For $i \in \{0,1,2\}$, $\widetilde{\text{VaR}}^i(1-\alpha) := \min_\gamma\{\gamma|\widetilde{Q}^i(\gamma) \leqslant \alpha\}$. $\widetilde{\text{VaR}}^0(1-\alpha) := \min_\gamma\{\gamma|\widetilde{Q}^0(\gamma) \leqslant \alpha\}$ is the optimal $\widetilde{\text{VaR}}(1-\alpha;\theta)$. By definition, $\widetilde{\text{VaR}}^2(1-\alpha)$ is the risk value of $\widetilde{\text{VaR}}(1-\alpha;\theta)$ with θ being the solutions of $\widetilde{\text{TiVaR}}(1-\alpha;\theta)$. $\widetilde{\text{VaR}}^{\widetilde{\text{CVaR}}}(1-\alpha)$ denotes the value of $\widetilde{\text{VaR}}(1-\alpha;\theta)$ evaluated at $\arg\min_\theta \widetilde{\text{CVaR}}(1-\alpha;\theta)$, and $\widetilde{\text{VaR}}^{\widetilde{\text{CVaR}}_{\text{inv}}}(1-\alpha)$ and $\widetilde{\text{VaR}}^{\widetilde{\text{EVaR}}}(1-\alpha)$ are defined in the similar way. We see that $\widetilde{\text{VaR}}^1(1-\alpha)$ and $\widetilde{\text{VaR}}^2(1-\alpha)$ are close to $\widetilde{\text{VaR}}^0(1-\alpha)$, which indicates VaR with the solutions obtained from solving $\widetilde{\text{TiVaR}}(1-\alpha;\theta)$ (which is $\widetilde{\text{VaR}}^2(1-\alpha)$) is a tight upper bound of the globally optimal $\widetilde{\text{VaR}}(1-\alpha;\theta)$. $\widetilde{\text{VaR}}^2(1-\alpha)$ is also tighter than VaR under EVaR solutions when α is not small.

Inverse CVaR. We note that while the most popular form of CVaR focuses on upper quantiles (as discussed in the main text), one may explore 'inverse' CVaR that can focus on lower quantiles, with its empirical form $\widetilde{\text{CVaR}}_{\text{inv}}(1-\alpha;\theta)$ for $\alpha \in [0,1)$ defined as

$$\widetilde{\text{CVaR}}_{\text{inv}}(1-\alpha;\theta) := -\min_\gamma \left\{ \gamma + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i\in[N]} [-f(x_i;\theta)-\gamma]_+ \right\}. \tag{7.196}$$

As α ranges from 0 to 1, optimizing $\widetilde{\text{CVaR}}_{\text{inv}}(1-\alpha;\theta)$ transitions from solving avg-loss to min-loss. However, different from TiVaR or CVaR, CVaR_{inv} is not a valid upper bound of VaR. Despite this, we optimize $\min_\theta \widetilde{\text{CVaR}}_{\text{inv}}(1-\alpha;\theta)$, plug in the optimal model parameters to evaluate VaR values, and compare with the approximate VaR values under the solutions of other risks including TiVaR. From Figure 7.16, we see that VaR values under TiVaR solutions can be smaller than those under CVaR_{inv} solutions on linear regression. Given any α, our proposed TiVaR objective approximates VaR, ranging from min-loss to max-loss smoothly *in a single formulation*, which can be more desirable than optimizing two objectives.

Proof of Theorem 28. We first prove $\widetilde{\mathrm{TiVaR}}(1-\alpha;\theta) \leqslant \widetilde{\mathrm{EVaR}}(1-\alpha;\theta)$.

$$\widetilde{\mathrm{EVaR}}(1-\alpha;\theta) - \widetilde{F}(-\infty) = \min_{t\in\mathbb{R}^{>0}} \frac{1}{t}\log\left(\frac{\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}}{\alpha}\right) - \widetilde{F}(-\infty) \tag{7.197}$$

$$= \min_{t\in\mathbb{R}^{>0}} \frac{1}{t}\log\left(\frac{e^{(\widetilde{R}(t;\theta)-\widetilde{F}(-\infty))t}}{\alpha}\right) \tag{7.198}$$

$$\geqslant \min_{t\in\mathbb{R}^{>0}} \frac{1}{t}\log\left[\frac{e^{(\widetilde{R}(t;\theta)-\widetilde{F}(-\infty))t} - (1-\alpha)}{\alpha}\right]_+ \tag{7.199}$$

$$\geqslant \min_{t\in\mathbb{R}} \frac{1}{t}\log\left[\frac{e^{(\widetilde{R}(t;\theta)-\widetilde{F}(-\infty))t} - (1-\alpha)}{\alpha}\right]_+. \tag{7.200}$$

We next prove $\widetilde{\mathrm{VaR}}(1-\alpha;\theta) \leqslant \widetilde{\mathrm{TiVaR}}(1-\alpha;\theta)$. From Theorem 36, we know that for any t,θ,

$$\widetilde{Q}(\gamma;\theta) \leqslant \min_{t\in\mathbb{R}} \left\{\frac{e^{\widetilde{R}(t;\theta)t} - e^{-\widetilde{F}(-\infty)t}}{e^{\gamma t} - e^{\widetilde{F}(-\infty)t}}\right\} \tag{7.201}$$

Let $\widetilde{Q}(\gamma;\theta) = \alpha$, and $\gamma^* = \widetilde{\mathrm{VaR}}(1-\alpha;\theta)$. We have $\min_{t\in\mathbb{R}}\left\{\frac{e^{\widetilde{R}(t;\theta)t}-e^{-\widetilde{F}(-\infty)t}}{e^{\gamma^* t}-e^{\widetilde{F}(-\infty)t}}\right\} \geqslant \alpha$. We also note

$$\min_{t\in\mathbb{R}} \left\{\frac{e^{\widetilde{R}(t;\theta)t} - e^{-\widetilde{F}(-\infty)t}}{e^{\widetilde{\mathrm{TiVaR}}(1-\alpha;\theta)t} - e^{\widetilde{F}(-\infty)t}}\right\} = \alpha. \tag{7.202}$$

Hence,

$$\widetilde{\mathrm{TiVaR}}(1-\alpha;\theta) \geqslant \gamma^* = \widetilde{\mathrm{VaR}}(1-\alpha;\theta). \tag{7.203}$$

TERM and Entropic Value-at-Risk. Let $\breve{\theta}_X(t)$ be the minimizer of entropic risk $R_X(t;\theta)$:

$$\breve{\theta}_X(t) := \arg\min_{\theta\in\Theta} R_X(t;\theta). \tag{7.204}$$

Further, let $F_X(t)$ be the optimum value of entropic risk, i.e.,

$$F_X(t) := R_X(t;\breve{\theta}_X(t)). \tag{7.205}$$

Our next result will relate EVaR to entropic risk.

Lemma 46 (Relations between entropic risk and EVaR)**.** *Assume that for $t \in \mathbb{R}^{>0}$, $F_X(t)$ is a strongly convex function of $\frac{1}{t}$. Further, let*

$$\breve{t}_X(\alpha) \in \arg\min_{t\in\mathbb{R}^{>0}} \left\{F_X(t) - \frac{1}{t}\log\alpha\right\}, \tag{7.206}$$

then

$$\arg\min_{\theta}\{EVaR_X(1-\alpha;\theta)\} = \arg\min_{\theta}\left\{R_X(\check{t}_X(\alpha);\theta)\right\} := \check{\theta}_X(\check{t}_X(\alpha)), \tag{7.207}$$

$$R_X\left(\check{t}_X(\alpha);\check{\theta}_X(\check{t}_X(\alpha))\right) = F_X(\check{t}_X(\alpha)) \leqslant EVaR_X\left(1-\alpha;\check{\theta}_X(\check{t}_X(\alpha))\right). \tag{7.208}$$

Proof. Consider the any minimizer of $R_X(\check{t}_X(\alpha),\theta)$, i.e.,

$$\check{\theta}_X(\check{t}_X(\alpha)) \in \arg\min_{\theta} R_X(\check{t}_X(\alpha);\theta), \tag{7.209}$$

we next prove

$$\check{\theta}_X(\check{t}_X(\alpha)) \in \arg\min_{\theta}\left(\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\theta)}] - \frac{1}{t}\log\alpha\right)\right). \tag{7.210}$$

Denote $\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\theta)}] - \frac{1}{t}\log\alpha\right)$ as $h(\theta;\alpha)$. Let

$$\theta_v^* \in \arg\min_{\theta} h(\theta;\alpha), \tag{7.211}$$

$$t_v(\theta_v^*) \in \arg\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\theta_v^*)}] - \frac{1}{t}\log\alpha\right). \tag{7.212}$$

By the definition of $\check{t}_X(\alpha)$ and $\check{\theta}_X(t)$, we have

$$\frac{1}{\check{t}_X(\alpha)}\log\mathbb{E}[e^{\check{t}_X(\alpha)f(X;\check{\theta}_X(\check{t}_X(\alpha)))}] - \frac{1}{\check{t}_X(\alpha)}\log\alpha \tag{7.213}$$

$$\leqslant \frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\check{\theta}_X(t_v(\theta_v^*)))}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.214}$$

$$\leqslant \frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\theta_v^*)}] - \frac{1}{t_v(\theta_v^*)}\log\alpha. \tag{7.215}$$

By the definition of θ_v^*, $h(\theta_v^*;\alpha) \leqslant h(\check{\theta}_X(\check{t}_X(\alpha));\alpha)$, i.e.,

$$\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\theta_v^*)}] - \frac{1}{t}\log\alpha\right) \leqslant \min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\check{\theta}_X(\check{t}_X(\alpha)))}] - \frac{1}{t}\log\alpha\right). \tag{7.216}$$

We have

$$\frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\theta_v^*)}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.217}$$

$$\leqslant \min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\check{\theta}_X(\check{t}_X(\alpha)))}] - \frac{1}{t}\log\alpha\right) \tag{7.218}$$

$$\leqslant \frac{1}{\check{t}_X(\alpha)}\log\mathbb{E}[e^{\check{t}_X(\alpha)f(X;\check{\theta}_X(\check{t}_X(\alpha)))}] - \frac{1}{\check{t}_X(\alpha)}\log\alpha, \tag{7.219}$$

Hence, $\check{\theta}_X(\check{t}_X(\alpha)) \in \arg\min_{\theta}\left(\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\theta)}] - \frac{1}{t}\log\alpha\right)\right)$.

For the other direction, consider any minimizer of $\mathrm{EVaR}_X(1-\alpha;\theta)$, i.e.,

$$\theta_v^* \in \arg\min_\theta h(\theta;\alpha) \tag{7.220}$$

$$t_v(\theta_v^*) \in \arg\min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{f(X;\theta_v^*)}] - \frac{1}{t}\log\alpha\right). \tag{7.221}$$

We next prove $\theta_v^* \in \arg\min_\theta \frac{1}{\breve{t}_X(\alpha)}\log\mathbb{E}[e^{\breve{t}_X(\alpha)f(X;\theta)}]$. By the definition of $\breve{\theta}_X(t)$ and $\breve{t}_X(\alpha)$,

$$\frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\theta_v^*)}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.222}$$

$$\geqslant \frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\breve{\theta}_X(t_v(\theta_v^*)))}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.223}$$

$$\geqslant \frac{1}{\breve{t}_X(\alpha)}\log\mathbb{E}[e^{\breve{t}_X(\alpha)f(X;\breve{\theta}_X(\breve{t}_X(\alpha)))}] - \frac{1}{\breve{t}_X(\alpha)}\log\alpha. \tag{7.224}$$

On the other hand, by the definition of θ_v^* and $t_v(\theta_v^*)$,

$$\frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\theta_v^*)}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \leqslant \min_{t>0}\left(\frac{1}{t}\log\mathbb{E}[e^{tf(X;\breve{\theta}_X(\breve{t}_X(\alpha)))}] - \frac{1}{t}\log\alpha\right) \tag{7.225}$$

$$\leqslant \frac{1}{\breve{t}_X(\alpha)}\log\mathbb{E}[e^{\breve{t}_X(\alpha)f(X;\breve{\theta}_X(\breve{t}_X(\alpha)))}] - \frac{1}{\breve{t}_X(\alpha)}\log\alpha. \tag{7.226}$$

Therefore,

$$\frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\theta_v^*)}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.227}$$

$$= \frac{1}{t_v(\theta_v^*)}\log\mathbb{E}[e^{t_v(\theta_v^*)f(X;\breve{\theta}_X(t_v(\theta_v^*)))}] - \frac{1}{t_v(\theta_v^*)}\log\alpha \tag{7.228}$$

$$= \frac{1}{\breve{t}_X(\alpha)}\log\mathbb{E}[e^{\breve{t}_X(\alpha)f(X;\breve{\theta}_X(\breve{t}_X(\alpha)))}] - \frac{1}{\breve{t}_X(\alpha)}\log\alpha. \tag{7.229}$$

If

$$s(t) := \frac{1}{t}\log\mathbb{E}[e^{tf(X;\breve{\theta}_X(t))}] - \frac{1}{t}\log\alpha \tag{7.230}$$

has a unique minimizer,

$$\breve{t}_X(\alpha) = t_v(\theta_v^*), \tag{7.231}$$

and

$$\theta_v^* \in \arg\min_\theta \frac{1}{\breve{t}_X(\alpha)}\log\mathbb{E}[e^{\breve{t}_X(\alpha)f(X;\theta)}]. \tag{7.232}$$

Hence, we have proved

$$\arg\min_{\theta} \mathrm{EVaR}_X(1-\alpha;\theta) = \arg\min_{\theta} R_X(\check{t}_X(\alpha);\theta), \tag{7.233}$$

and

$$R_X(\check{t}_X(\alpha);\check{\theta}_X(\check{t}_X(\alpha))) \leqslant \mathrm{EVaR}_X(1-\alpha;\check{\theta}_X(\check{t}_X(\alpha))). \tag{7.234}$$

□

The lemma relates the solution and the optimal value of EVaR with those of entropic risk. We can extend Lemma 46 to the empirical version below.

Lemma 47 (Relations between empirical entropic risk and empirical EVaR)**.** *Assume that* $\widetilde{F}(t)$ *is a strongly convex function of* $\frac{1}{t}$*. For* $\alpha \in \{\frac{k}{N}\}_{k\in[N]}$*, let*

$$\check{t}(\alpha) \in \arg\min_{t>0} \left\{ \widetilde{F}(t) - \frac{1}{t}\log\alpha \right\}, \tag{7.235}$$

then

$$\arg\min_{\theta} \widetilde{EVaR}(1-\alpha;\theta) = \arg\min_{\theta} \widetilde{R}(\check{t}(\alpha);\theta), \tag{7.236}$$

$$\widetilde{F}(\check{t}(\alpha)) \leqslant \widetilde{EVaR}(1-\alpha;\check{\theta}(\check{t}(\alpha))). \tag{7.237}$$

7.11 Solving TERM (Proofs and Details)

7.11.1 Hierarchical Multi-Objective Tilting

We state the hierarchical multi-objective tilting for a hierarchy of depth 3. While we don't directly use this form, it is stated to clarify the experiments in Section 7.7 where tilting is done at class level and annotator level, and the sample-level tilt value could be understood to be 0.

$$\widetilde{J}(m,t,\tau;\theta) := \frac{1}{m}\log\left(\frac{1}{N}\sum_{G\in[GG]}\left(\sum_{g\in[G]}|g|\right)e^{m\widetilde{J}_G(\tau;\theta)}\right) \tag{7.238}$$

$$\widetilde{J}_G(t,\tau;\theta) := \frac{1}{t}\log\left(\frac{1}{\sum_{g\in[G]}|g|}\sum_{g\in[G]}|g|e^{t\widetilde{R}_g(\tau;\theta)}\right) \tag{7.239}$$

$$\widetilde{R}_g(\tau;\theta) := \frac{1}{\tau}\log\left(\frac{1}{|g|}\sum_{x\in g}e^{\tau f(x;\theta)}\right), \tag{7.240}$$

Proof of Lemma 31. We proceed as follows. First notice that by invoking Lemma 26,

$$\nabla_\theta\widetilde{J}(t,\tau;\theta) = \sum_{g\in[G]} w_g(t,\tau;\theta)\nabla_\theta\widetilde{R}_g(\tau;\theta) \tag{7.241}$$

where

$$w_g(t,\tau;\theta) := \frac{|g|e^{t\widetilde{R}_g(\tau;\theta)}}{\sum_{g'\in[G]}|g'|e^{t\widetilde{R}_{g'}(\tau;\theta)}}. \tag{7.242}$$

where $\widetilde{R}_g(\tau;\theta)$ is defined in (7.82), and is reproduced here:

$$\widetilde{R}_g(\tau;\theta) := \frac{1}{\tau}\log\left(\frac{1}{|g|}\sum_{x\in g}e^{\tau f(x;\theta)}\right). \tag{7.243}$$

On the other hand, by invoking Lemma 26,

$$\nabla_\theta\widetilde{R}_g(\tau;\theta) = \sum_{x\in g} w_{g,x}(\tau;\theta)\nabla_\theta f(x;\theta) \tag{7.244}$$

where

$$w_{g,x}(\tau;\theta) := \frac{e^{\tau f(x;\theta)}}{\sum_{y\in g}e^{\tau f(y;\theta)}}. \tag{7.245}$$

Hence, combining (7.241) and (7.244),

$$\nabla_\theta\widetilde{J}(t,\tau;\theta) = \sum_{g\in[G]}\sum_{x\in g} w_g(t,\tau;\theta)w_{g,x}(\tau;\theta)\nabla_\theta f(x;\theta). \tag{7.246}$$

The proof is completed by algebraic manipulations to show that

$$w_{g,x}(t,\tau;\theta) = w_g(t,\tau;\theta) w_{g,x}(\tau;\theta). \tag{7.247}$$

□

7.11.2 Proofs of Convergence for TERM Solvers

Algorithm 17 Stochastic Non-Hierarchical TERM with two mini-batches

Input: θ, $\widetilde{\widetilde{R}}_t = \frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta)}\right)$, t, α, λ
for iter $= 0, \cdots, T-1$ **do**
  sample two independent minibatches B_1, B_2 uniformly at random from $[N]$
  compute the loss $f(x;\theta)$ and gradient $\nabla_\theta f(x;\theta)$ for all $x \in B_1$
  $\widetilde{R}_{B,t} \leftarrow t$-tilted loss (7.2) on minibatch B_2
  $\widetilde{\widetilde{R}}_t \leftarrow \frac{1}{t}\log\left((1-\lambda)e^{t\widetilde{\widetilde{R}}_t} + \lambda e^{t\widetilde{R}_{B,t}}\right)$ $w_{t,x} \leftarrow e^{tf(x;\theta) - t\widetilde{\widetilde{R}}_t}$
  $\theta \leftarrow \theta - \frac{\alpha}{|B_1|}\sum_{x\in B_1} w_{t,x}\nabla_\theta f(x;\theta)$
end for

To prove our convergence results in Theorem 32, we first prove a lemma below.

Lemma 48. *Denote* $k_t := \arg\max_k \left(k < \frac{2e}{\mu} + \frac{etLBe^{t(\tilde{F}_{\max} - \tilde{F}_{\min})}}{\mu k}\right)$. *Let* $\lambda = 1 - \frac{1}{2e}$, *and*

$$\alpha_k = \begin{cases} \frac{1}{tLBe^{t(\tilde{F}_{\max} - \tilde{F}_{\min})}+1}, & \text{if } k \leqslant k_t \\ \frac{2e}{\mu k}, & \text{otherwise,} \end{cases} \tag{7.248}$$

then for any k,

$$\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k] \leqslant 2e, \tag{7.249}$$

where $\widetilde{R}_k := \widetilde{R}(t;\theta_k) = \frac{1}{t}\log\left(\frac{1}{N}\sum_{i\in[N]} e^{tf(x_i;\theta_k)}\right)$.

Proof. We have the updating rule

$$e^{t\widetilde{\widetilde{R}}_{k+1}} = \lambda e^{tf(\xi_k,\theta_k)} + (1-\lambda)e^{t\widetilde{\widetilde{R}}_k}. \tag{7.250}$$

Taking conditional expectation $\mathbb{E}[\cdot|\theta_1, \ldots, \theta_{k+1}]$ on both sides of (7.250) gives

$$\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_k)} | \theta_1, \ldots, \theta_{k+1}] \tag{7.251}$$

$$= \lambda \mathbb{E}[e^{t(f(\xi_k;\theta_k) - \widetilde{R}_k)} | \theta_1, \ldots, \theta_{k+1}] + (1-\lambda)\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_{k+1}] \tag{7.252}$$

$$= \lambda + (1-\lambda)\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k]. \tag{7.253}$$

For any k, we have

$$\|\theta_{k+1} - \theta_k\| = \alpha_k \left\| \frac{e^{t\widetilde{R}_k}}{e^{t\widetilde{\widetilde{R}}_k}} \nabla \widetilde{R}_k \right\| \leqslant \alpha_k e^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} B. \tag{7.254}$$

Therefore,

$$|f(x_i; \theta_{k-1}) - f(x_i; \theta_k)| \leqslant L\|\theta_{k-1} - \theta_k\| \leqslant \alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}, \tag{7.255}$$

and

$$e^{-t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \leqslant e^{t(\widetilde{R}_k - \widetilde{R}_{k+1})} = \frac{\sum_{i\in[N]} e^{tf(x_i;\theta_k)}}{\sum_{i\in[N]} e^{tf(x_i;\theta_{k+1})}} \leqslant e^{t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}, \tag{7.256}$$

$$\begin{aligned}
& e^{-t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_{k+1})} | \theta_1, \ldots, \theta_{k+1}] && (7.257)\\
& \leqslant \mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_k)} | \theta_1, \ldots, \theta_{k+1}] \\
& \leqslant e^{t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_{k+1})} | \theta_1, \ldots, \theta_{k+1}]. && (7.258)
\end{aligned}$$

Hence,

$$\begin{aligned}
& e^{-t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_{k+1})} | \theta_1, \ldots, \theta_{k+1}] && (7.259)\\
& \leqslant \lambda + (1-\lambda)\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k] && (7.260)\\
& \leqslant e^{t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k+1} - \widetilde{R}_{k+1})} | \theta_1, \ldots, \theta_{k+1}]. && (7.261)
\end{aligned}$$

(i) When $k \leqslant k_t$, under the learning rate α_k set as in Eq. (7.248), we have

$$\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} < 1. \tag{7.262}$$

Hence,

$$\begin{aligned}
\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k] & \leqslant e(\lambda + (1-\lambda)\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k-1} - \widetilde{R}_{k-1})} | \theta_1, \ldots, \theta_{k-1}]) && (7.263)\\
& \leqslant e + \frac{1}{2}\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k-1} - \widetilde{R}_{k-1})} | \theta_1, \ldots, \theta_{k-1}] && (7.264)\\
& \leqslant \cdots \leqslant e\left(2 - \frac{1}{2^{k-2}}\right) + \frac{1}{2^{k-1}}\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_1 - \widetilde{R}_1)} | \theta_1] \leqslant 2e. && (7.265)
\end{aligned}$$

(ii) When $k > k_t$,

$$\alpha_k = \frac{2e}{\mu k} < \frac{k}{k + tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}. \tag{7.266}$$

Similarly, we have

$$\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k] \leqslant e^{t\alpha_k LBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}} \left(\lambda + (1-\lambda)\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_{k-1} - \widetilde{R}_{k-1})} | \theta_1, \ldots, \theta_{k-1}]\right) \tag{7.267}$$

$$\leqslant \cdots \leqslant 2e, \tag{7.268}$$

which completes the proof. □

Proof of Theorem 32 Denote the empirical optimal solution $\breve{\theta}(t)$ as θ^*. Denote the tilted stochastic gradient on data ζ_k as g_k, where

$$g_k = \frac{e^{tf(\zeta_k;\theta_k)}}{e^{t\widetilde{\widetilde{R}}_k}} \nabla f(\zeta_k;\theta_k) = \frac{e^{t\widetilde{R}_k}}{e^{t\widetilde{\widetilde{R}}_k}} \frac{e^{tf(\zeta_k;\theta_k)}}{e^{t\widetilde{R}_k}} \nabla f(\zeta_k;\theta_k) = \frac{e^{t\widetilde{R}_k}}{e^{t\widetilde{\widetilde{R}}_k}} \nabla \widetilde{R}_k(\zeta_k). \tag{7.269}$$

Therefore, for any $k \geqslant 1$,

$$\begin{aligned}
\mathbb{E}[\langle \theta_k - \theta^*, g_k \rangle] &= \mathbb{E}[\mathbb{E}[\langle \theta_k - \theta^*, g_k \rangle | \theta_1, \ldots, \theta_k]] && (7.270)\\
&= \mathbb{E}[\langle \theta_k - \theta^*, \mathbb{E}[g_k | \theta_1, \ldots, \theta_k] \rangle] && (7.271)\\
&= \mathbb{E}[\langle \theta_k - \theta^*, \mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} | \theta_1, \ldots, \theta_k] \mathbb{E}[\nabla \widetilde{R}_k(\zeta_k) | \theta_1, \ldots, \theta_k] \rangle] && (7.272)\\
&\geqslant \frac{1}{2e} \mathbb{E}[\langle \theta_k - \theta^*, \nabla \widetilde{R}(\theta_k) \rangle] \quad (\mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} | \theta_1, \ldots, \theta_k] \geqslant 1/\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k]) && (7.273)\\
&\geqslant \frac{\mu}{2e} \mathbb{E}[\|\theta_k - \theta^*\|^2] \quad (\mu\text{-strong convexity of } \widetilde{R}), && (7.274)
\end{aligned}$$

where (7.272) follows from the fact that $e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)}$ and $\nabla \widetilde{R}_k(\zeta_k)$ are independent given $\{\theta_1, \ldots, \theta_k\}$. For $k \geqslant k_t$ with $\alpha_k = \frac{2e}{\mu k}$,

$$\begin{aligned}
\mathbb{E}[\|\theta_{k+1} - \theta^*\|^2] &= \mathbb{E}[\|\theta_k - \alpha_k g_k - \theta^*\|^2] && (7.275)\\
&= \mathbb{E}[\|\theta_k - \theta^*\|^2] - 2\alpha_k \mathbb{E}[\langle \theta_k - \theta^*, g_k \rangle] + \alpha_k^2 \mathbb{E}[\|g_k\|^2] && (7.276)\\
&\leqslant \left(1 - \frac{\alpha_k \mu}{e}\right) \mathbb{E}[\|\theta_k - \theta^*\|^2] + \alpha_k^2 \mathbb{E}[\|e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k)\|^2] && (7.277)\\
&\leqslant \left(1 - \frac{2}{k}\right) \mathbb{E}[\|\theta_k - \theta^*\|^2] + \frac{4e^2 B^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{\mu^2 k^2}. && (7.278)
\end{aligned}$$

When $k \leqslant k_t$ with $\alpha_k = \frac{1}{1 + tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}$,

$$\mathbb{E}[\|\theta_k - \theta^*\|^2] \leqslant \left(1 - \frac{\mu}{e(tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} + 1)}\right) \mathbb{E}[\|\theta_{k-1} - \theta^*\|^2] + \frac{B^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{(1 + tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})})^2}.$$

We can thus prove

$$\mathbb{E}[\|\theta_{k_t} - \theta^*\|^2] \leqslant \max\left\{\mathbb{E}[\|\theta_1 - \theta^*\|^2], \frac{B^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min}) + 1}}{\mu(1 + tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})})}\right\} \tag{7.279}$$

Let

$$V_t = \max\left\{k_t \mathbb{E}[\|\theta_{k_t} - \theta^*\|^2], \frac{4B^2 e^{2+2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{\mu^2}\right\}. \tag{7.280}$$

We next prove for $k \geqslant k_t$,

$$\mathbb{E}[\|\theta_k - \theta^*\|^2] \leqslant \frac{V_t}{k}. \tag{7.281}$$

Suppose $\mathbb{E}[\|\theta_k - \theta^*\|^2] \leqslant \frac{V_t}{k}$. From (7.278), we have

$$\mathbb{E}[\|\theta_{k+1} - \theta^*\|^2] \leqslant \left(1 - \frac{2}{k}\right) \mathbb{E}[\|\theta_k - \theta^*\|^2] + \frac{4e^2B^2e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{k^2\mu^2} \tag{7.282}$$

$$\leqslant \left(1 - \frac{2}{k}\right) \frac{V_t}{k} + \frac{V_t^2}{k^2} \tag{7.283}$$

$$\leqslant \frac{V_t}{k+1}, \tag{7.284}$$

where $k \geqslant k_t = \left\lceil \frac{e + \sqrt{e^2 + \mu t L B e^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} + 1}}{\mu} \right\rceil$. This completes the proof. □

Proof of Theorem 33. Assume $\widetilde{R}(t;\theta)$ is non-convex and β-smooth, we have

$$\widetilde{R}_{k+1} - \widetilde{R}_k - \langle \nabla \widetilde{R}_k, \theta_{k+1} - \theta_k \rangle \leqslant \frac{\beta}{2} \|\theta_{k+1} - \theta_k\|^2, \tag{7.285}$$

where $\widetilde{R}_k := \widetilde{R}(t;\theta_k) = \frac{1}{t} \log \left(\frac{1}{N} \sum_{i \in [N]} e^{t f(x_i;\theta_k)} \right)$. Plugging in the updating rule

$$\theta_{k+1} - \theta_k = -\alpha_k \frac{e^{t(\zeta_k;\theta_k)}}{e^{t\widetilde{\widetilde{R}}_k}} \nabla f(\zeta_k;\theta_k) = -\alpha_k e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \tag{7.286}$$

gives

$$\widetilde{R}_{k+1} - \widetilde{R}_k + \alpha_k \langle \nabla \widetilde{R}_k, e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \rangle \leqslant \frac{\beta}{2} \left\| \alpha_k e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \right\|^2. \tag{7.287}$$

First, we note

$$\left\| \alpha_k^2 e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \right\|^2 \leqslant \alpha_k^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} \|\nabla \widetilde{R}_k(\zeta_k)\|^2. \tag{7.288}$$

Take expectation on both sides of (7.287),

$$\mathbb{E}[\widetilde{R}_{k+1}] - \mathbb{E}[\widetilde{R}_k] + \alpha_k \mathbb{E}[\langle \nabla \widetilde{R}_k, e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \rangle] \leqslant \frac{\beta \alpha_k^2 e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} B^2}{2}. \tag{7.289}$$

Let

$$k_t := \left\lceil \frac{2(\widetilde{F}_{\max} - \widetilde{F}_{\min}) t^2 L^2}{\beta e^2} \right\rceil. \tag{7.290}$$

For any $k \geqslant k_t$, let

$$\alpha_k = \frac{\sqrt{2(\widetilde{F}_{\max} - \widetilde{F}_{\min})}}{e^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})}\sqrt{\beta B^2 K}}. \tag{7.291}$$

For $k < k_t$, let

$$\alpha_k = \frac{1}{tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} + 1}. \tag{7.292}$$

We have for any $k \geqslant 1$,

$$\alpha_k tLBe^{t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} \leqslant 1. \tag{7.293}$$

Therefore, for any $k \geqslant 1$,

$$\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k - \widetilde{R}_k)} | \theta_1, \ldots, \theta_k] \leqslant 2e. \tag{7.294}$$

Thus, for any $k \geqslant 1$,

$$\begin{aligned}
\mathbb{E}[\langle \nabla \widetilde{R}_k, e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \rangle] &= \mathbb{E}[\mathbb{E}[\langle \nabla \widetilde{R}_k, e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) \rangle | \theta_1, \ldots, \theta_k]] && (7.295)\\
&= \mathbb{E}[\langle \nabla \widetilde{R}_k, \mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} \nabla \widetilde{R}_k(\zeta_k) | \theta_1, \ldots, \theta_k] \rangle] && (7.296)\\
&= \mathbb{E}[\langle \nabla \widetilde{R}_k, \mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} | \theta_1, \ldots, \theta_k] \mathbb{E}[\nabla \widetilde{R}_k(\zeta_k) | \theta_1, \ldots, \theta_k] \rangle] && (7.297)\\
&= \mathbb{E}[\langle \nabla \widetilde{R}_k, \mathbb{E}[e^{t(\widetilde{R}_k - \widetilde{\widetilde{R}}_k)} | \theta_1, \ldots, \theta_k] \nabla \widetilde{R}_k \rangle] && (7.298)\\
&\geqslant \frac{1}{2e} \mathbb{E}[\|\nabla \widetilde{R}_k\|^2]. && (7.299)
\end{aligned}$$

Plug (7.299) into (7.289),

$$\mathbb{E}[\|\nabla \widetilde{R}_k\|^2] + \frac{2e}{\alpha_k}(\mathbb{E}[\widetilde{R}_{k+1}] - \mathbb{E}[\widetilde{R}_k]) \leqslant \beta \alpha_k e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} eB^2. \tag{7.300}$$

Apply telescope sum from $k_t + 1$ to K and divide both sides by K,

$$\frac{1}{K} \sum_{k=k_t}^{K} \mathbb{E}[\|\nabla \widetilde{R}_k\|^2] + \frac{2e(\mathbb{E}[\widetilde{R}_{K+1}] - \mathbb{E}[\widetilde{R}_{k_t}])}{\alpha_k K} \leqslant \beta \alpha_k e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} eB^2. \tag{7.301}$$

$$\begin{aligned}
\frac{1}{K} \sum_{k=k_t}^{K} \mathbb{E}[\|\nabla \widetilde{R}_k\|^2] &\leqslant \beta \alpha_k e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} eB^2 + \frac{2e(\mathbb{E}[\widetilde{R}_{k_t} - \widetilde{R}_{K+1}])}{\alpha_k K} && (7.302)\\
&\leqslant \beta \alpha_k e^{2t(\widetilde{F}_{\max} - \widetilde{F}_{\min})} eB^2 + \frac{2e(\widetilde{F}_{\max} - \widetilde{F}_{\min})}{\alpha_k K} && (7.303)
\end{aligned}$$

Consider that $\alpha_k = \frac{\sqrt{2(\widetilde{F}_{\max}-\widetilde{F}_{\min})}}{e^{t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}\sqrt{\beta B^2 K}}$,

$$\frac{1}{K}\sum_{k=k_t}^{K}\mathbb{E}[\|\nabla \widetilde{R}_k\|^2] \leqslant \sqrt{8}Be^{t(\widetilde{F}_{\max}-\widetilde{F}_{\min})+1}\sqrt{\frac{\beta(\widetilde{F}_{\max}-\widetilde{F}_{\min})}{K}}, \tag{7.304}$$

completing the proof. □

Proof of Theorem 34. From the assumptions, we have $\widetilde{R}(t;\theta)$ is $\frac{\mu}{2}$-PL, i.e.,

$$\mu(\widetilde{R}(t;\theta)-\widetilde{R}^*) \leqslant \|\nabla \widetilde{R}(t;\theta)\|^2, \tag{7.305}$$

where $\widetilde{R}^* := \widetilde{R}(t;\breve{\theta}(t))$. Let

$$k_t := \arg\max_k \left(k < \frac{4e}{\mu} + \frac{4etLBe^{t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}}{\mu k} \right), \tag{7.306}$$

and

$$\alpha_k = \begin{cases} \frac{1}{tLBe^{t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}+1}, & \text{if } k \leqslant k_t \\ \frac{4e}{\mu k}, & \text{otherwise.} \end{cases} \tag{7.307}$$

Similarly, we can prove for any $k \geqslant 1$,

$$\mathbb{E}[e^{t(\widetilde{\widetilde{R}}_k-\widetilde{R}_k)}|\theta_1,\ldots,\theta_k] \leqslant 2e. \tag{7.308}$$

Similarly,

$$\mathbb{E}[\widetilde{R}_{k+1}]-\mathbb{E}[\widetilde{R}_k]+\frac{\alpha_k}{2e}\mathbb{E}[\|\nabla \widetilde{R}_k\|^2] \leqslant \frac{\beta\alpha_k^2 e^{2t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}B^2}{2}. \tag{7.309}$$

Therefore,

$$\mathbb{E}[\widetilde{R}_{k+1}]-\mathbb{E}[\widetilde{R}_k]+\frac{\alpha_k}{2e}\mu\mathbb{E}[\widetilde{R}_k-\widetilde{R}^*] \leqslant \frac{\beta\alpha_k^2 e^{2t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}B^2}{2} \tag{7.310}$$

$$\mathbb{E}[\widetilde{R}_{k+1}-\widetilde{R}^*]-\mathbb{E}[\widetilde{R}_k-\widetilde{R}^*]+\frac{\alpha_k}{2e}\mu\mathbb{E}[\widetilde{R}_k-\widetilde{R}^*] \leqslant \frac{\beta\alpha_k^2 e^{2t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}B^2}{2} \tag{7.311}$$

$$\mathbb{E}[\widetilde{R}_{k+1}-\widetilde{R}^*] \leqslant \left(1-\frac{\alpha_k}{2e}\mu\right)\mathbb{E}[\widetilde{R}_k-\widetilde{R}^*]+\frac{\beta\alpha_k^2 e^{2t(\widetilde{F}_{\max}-\widetilde{F}_{\min})}B^2}{2} \tag{7.312}$$

Let $\alpha_k = \frac{4e}{\mu k}$, and

$$V_t = \max\left\{ k_t\mathbb{E}[\widetilde{R}_{k_t}-\widetilde{R}^*], \frac{8\beta B^2 e^{2t(\widetilde{F}_{\max}-\widetilde{F}_{\min})+2}}{\mu^2} \right\}. \tag{7.313}$$

We next prove $\mathbb{E}[\widetilde{R}_k - \widetilde{R}^*] \leqslant \frac{1}{k}$ $(k \geqslant k_t)$ by induction. Suppose $\mathbb{E}[\widetilde{R}_k - \widetilde{R}^*] \leqslant \frac{V_t}{k}$, then

$$\mathbb{E}[\widetilde{R}_{k+1} - \widetilde{R}^*] \leqslant \left(1 - \frac{2}{k}\right) \mathbb{E}[\widetilde{R}_k - \widetilde{R}^*] + \frac{V_t}{k^2} \tag{7.314}$$

$$\leqslant \left(1 - \frac{2}{k}\right) \frac{V_t}{k} + \frac{V_t}{k^2} \tag{7.315}$$

$$\leqslant \frac{V_t}{k+1}, \tag{7.316}$$

which concludes the proof. □

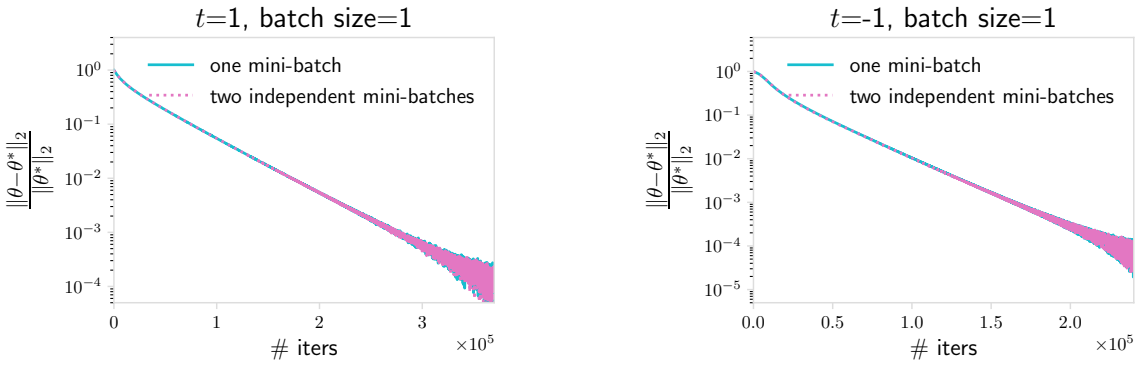


Figure 7.17: Convergence of Algorithm 14 using two independent mini-batches to update $\widetilde{\widetilde{R}}_t$ and calculate $e^{tf(x;\theta)}\nabla_\theta f(x;\theta)$ and a simpler variant using only one mini-batch to query $w_{t,x}\nabla_\theta f(x;\theta)$. We plot the optimality gap versus the number of iterations on the point estimation example (Figure 7.1 (a)) with batch size being 1. While Algorithm 14 allows us to get better convergence guarantees theoretically, we find that these two variants perform similarly empirically.

Table 7.10: TERM Applications and their corresponding solvers.

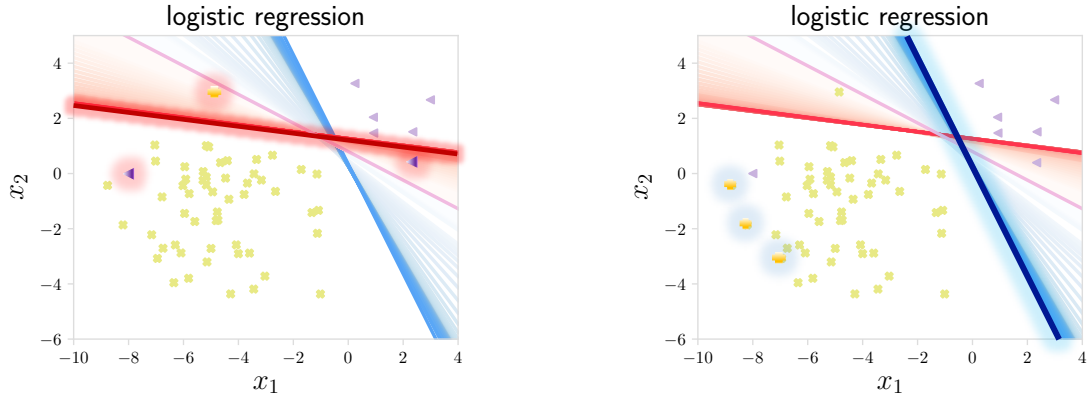
Three toy examples (Figure 7.1)	Algorithm 13
Robust regression (Table 7.2)	Algorithm 13
Robust classification (Table 7.4)	Algorithm 14
Low-quality annotators (Figure 7.9)	Algorithm 16 ($\tau = 0$)
Fair PCA (Figure 7.10)	Algorithm 15 ($\tau = 0$)
Class imbalance (Figure 7.14)	Algorithm 16 ($\tau = 0$)
Variance reduction (Table 7.7)	Algorithm 15 ($\tau = 0$)
Hierarchical TERM (Table 7.8)	Algorithm 15

7.12 Additional Experiments and Experimental Details

In Appendix 7.12.1, we provide complete experimental results on the properties or the use-cases of TERM. Details on how the experiments in Section 7.7 were executed are provided in Appendix 7.12.2.

7.12.1 Complete Results

Recall that in Section 7.2, Interpretation 1 is that TERM can be tuned to re-weight samples to magnify or suppress the influence of outliers. In Figure 7.18 below, we visually show this effect by highlighting the samples with the largest weight for $t \to +\infty$ and $t \to -\infty$ on the logistic regression example previously described in Figure 7.1.



(a) Samples with the largest weights as $t \to +\infty$. (b) Samples with the largest weights as $t \to -\infty$.

Figure 7.18: For positive values of t, TERM focuses on the samples with relatively large losses (rare instances). When $t \to +\infty$ (left), a few misclassified samples have the largest weights and are highlighted. On the other hand, for negative values of t, TERM suppresses the effect of the outliers, and as $t \to -\infty$ (right), samples with the smallest losses hold the the largest weights.

Next, we provide complete results of applying TERM to a diverse set of applications.

Robust classification. Recall that in Section 7.7.1, for classification in the presence of label noise, we only compare with baselines which do not require clean validation data. In Table 7.11 below, we report the complete results of comparing TERM with all baselines, including MentorNet-DD [130] which needs additional clean data. In particular, in contrast to the other methods, MentorNet-DD uses 5,000 clean validation images. TERM is competitive with the performance of MentorNet-DD, even though it does not have access to this clean data.

To interpret the noise more easily, we provide a toy logistic regression example with synthetic data here. In Figure 7.19, we see that TERM with $t = -2$ (blue) can converge to the correct classifier under 20%, 40%, and 80% noise.

Table 7.11: A complete comparison including two MentorNet variants. TERM is able to match the performance of MentorNet-DD, which needs additional clean labels.

objectives	**test accuracy** (`CIFAR-10`, Inception)		
	20% noise	40% noise	80% noise
ERM	0.775 (.004)	0.719 (.004)	0.284 (.004)
RandomRect [238]	0.744 (.004)	0.699 (.005)	0.384 (.005)
SelfPaced [159]	0.784 (.004)	0.733 (.004)	0.272 (.004)
MentorNet-PD [130]	0.798 (.004)	0.731 (.004)	0.312 (.005)
GCE [326]	**0.805** (.004)	0.750 (.004)	0.433 (.005)
MentorNet-DD [130]	**0.800** (.004)	**0.763** (.004)	**0.461** (.005)
TERM	0.795 (.004)	**0.768** (.004)	**0.455** (.005)
Genie ERM	0.828 (.004)	0.820 (.004)	0.792 (.004)

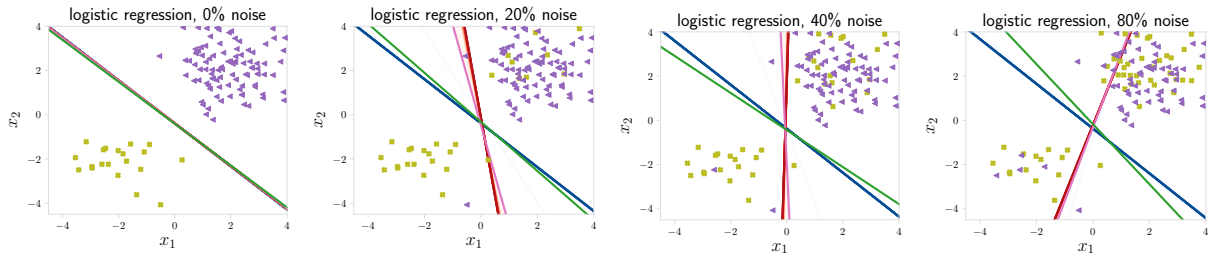


Figure 7.19: Robust classification using synthetic data. On this toy problem, we show that TERM with negative t's (blue) can be robust to random noisy samples. The green line corresponds to the solution of the generalized cross entropy (GCE) baseline [326]. Note that on this toy problem, GCE is as good as TERM with negative t's, despite its inferior performance on the real-world CIFAR10 dataset.

Low-quality annotators. In Section 7.7.1.3, we demonstrate that TERM can be used to mitigate the effect of noisy annotators, and we assume each annotator is either always correct, or always uniformly assigning random labels. Here, we explore a different and possibly more practical scenario where there are four noisy annotators who corrupt 0%, 20%, 40%, and 100% of their data by assigning labels uniformly at random, and there is one additional adversarial annotator who always assigns wrong labels. We assume the data points labeled by each annotator do not overlap, since [152] show that obtaining one label per sample is optimal for the data collectors under a fixed annotation budget. We compare TERM with several baselines: (a) training without the data coming from the adversarial annotator, (b) training without the data coming from the worst two annotators, and (c) training with all the clean data combined (Genie ERM). The results are shown in Figure 7.20. We see that TERM outperforms the strong baselines of removing one or two noisy annotators, and closely matches the performance of training with all the available clean data.

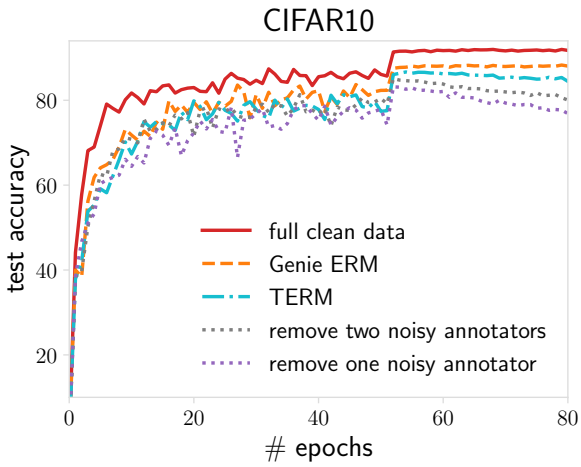


Figure 7.20: TERM achieves higher test accuracy than the baselines, and can match the performance of Genie ERM (i.e., training on all the clean data combined).

7.12.2 Experimental Details

We first describe the datasets and models used in each experiment presented in Section 7.7, and then provide a detailed setup including the choices of hyperparameters. All code and datasets are publicly available at github.com/litian96/TERM.

7.12.2.1 Datasets and Models

In Section 7.7.1, for regression tasks, we use the drug discovery data extracted from Diakonikolas et al. [71] which is originally curated from Olier et al. [224] and train linear regression models with different losses. There are 4,085 samples in total with each having 411 features. We randomly split the dataset into 80% training set, 10% validation set, and 10% testing set. For mitigating noise on classification tasks, we use the standard CIFAR-10 data and their standard train/val/test partitions along with a standard inception network [280]. For experiments regarding mitigating noisy annotators, we again use the CIFAR-10 data and their standard partitions with a ResNet20 model. The noise generation procedure is described in Section 7.7.1.3.

In Section 7.7.2, for fair PCA experiments, we use the complete Default Credit data to learn low-dimensional approximations and the loss is computed on the full training set. We follow the exact data processing steps described in the work [247] we compare with. There are 30,000 total data points with 21-dimensional features (after preprocessing). Among them, the high education group has 24,629 samples and the low education group has 5,371 samples. For meta-learning experiments, one the popular sine wave regression problem [92], we generate 5,000 meta-training and 5,000 meta-testing tasks. Following Collins et al. [57], there are 250 hard meta-training tasks with amplitudes drawn from $[4.95, 5]$ and 4,750 easy meta-training tasks with amplitudes drawn from $[0.01, 1]$. The amplitudes of meta-testing tasks are drawn uniformly from $[0.1, 5]$. The phase values are drawn uniformly from $[0, \pi]$ for all tasks. For class imbalance experiments, we directly

take the unbalanced data extracted from MNIST [165] used in Ren et al. [238]. When demonstrating the variance reduction of TERM, we use the HIV-1 dataset [245] as in [77] and randomly split it into 80% train, 10% validation, and 10% test set. There are 6,590 total samples and each has 160 features. We report results based on five such random partitions of the data. We train logistic regression models (without any regularization) for this binary classification task for TERM and the baseline methods. We also investigate the performance of a linear SVM.

In Section 7.7.3, the HIV-1 data are the same as that in Section 7.7.2. We also manually subsample the data to make it more imbalanced, or inject random noise, as described in Section 7.7.3. The CIFAR10 dataset used in this section is a standard benchmark, and we follow the same procedures in Cao et al. [45] to generate a noisy and imbalanced variant.

7.12.2.2 Hyperparameters

Selecting t. In Section 7.7.2 where we consider positive t's, we select t from a limited candidate set of $\{0.1, 1, 2, 5, 10, 50, 100, 200\}$ on the held-out validation set. For initial robust regression experiments, RMSE changed by only 0.08 on average across t; we thus used $t = -2$ for all experiments involving noisy training samples (Section 7.7.1 and Section 7.7.3).

Other Parameters. For all experiments, we tune all other hyperparameters (the learning rates, the regularization parameters, the decision threshold for ERM_+, ρ for [77], the quantile value for CVaR (i.e., α in Eq. (7.61)) [243], α and γ for focal loss [192]) based on a validation set, and select the best one. For experiments regarding focal loss [192], we select the class balancing parameter (α in the original focal loss paper) from $\texttt{range}(0.05, 0.95, 0.05)$ and select the main parameter γ from $\{0.5, 1, 2, 3, 4, 5\}$. We tune ρ in [77] such that $\frac{\rho}{n}$ is selected from $\{0.5, 1, 2, 3, 4, 5, 10\}$ where n is the training set size. We tune α for CVaR from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. All regularization parameters including regularization for linear SVM are selected from $\{0.0001, 0.01, 0.1, 1, 2\}$. For all experiments on the baseline methods, we use the default hyperparameters in the original paper (or the open-sourced code). We summarize a complete list of main hyperparameter values as follows.

Section 7.7.1:

- Robust regression. The threshold parameter δ for Huber loss for all noisy levels is 1, the corruption parameter k for CRR is: 500 (20% noise), 1000 (40% noise), and 3000 (80% noise); and TERM uses $t = -2$.
- Robust classification. The results are all based on the default hyperparameters provided by the open-sourced code of MentorNet [130], if applicable. We tune the q parameter for generalized cross entropy (GCE) from $\{0.4, 0.8, 1.0\}$ and select a best one for each noise level. For TERM, we scale t linearly as the number of iterations from 0 to -2 for all noise levels.
- Low-quality annotators. For all methods, we use the same set of hyperparameters. The initial step-size is set to 0.1 and decayed to 0.01 at epoch 50. The batch size is 100.

Section 7.7.2:

- Fair PCA. We use the default hyperparameters and directly run the public code of [247] to get the results on the min-max fairness baseline. We use a learning rate of 0.001 for our gradient-based solver for all target dimensions.
- Fair meta-learning. We use a fixed learning rate of 0.01 for all methods, and tune a best learning rate for the task weights for the work of Collins et al. [57]. Similar as Collins et al. [57], for all methods, we run one step of mini-batch SGD for inner optimization.
- Handling class imbalance. We take the open-sourced code of LearnReweight [238] and use the default hyperparameters for the baselines of LearnReweight, HardMine, and ERM. We implement focal loss, and select $\alpha = 0.05, \gamma = 2$.
- Variance reduction. The regularization parameter for linear SVM is 1. γ for focal loss is 2. We perform binary search on the decision thresholds for ERM_+ and RobustRegRisk_+, and choose 0.26 and 0.49, respectively.

Section 7.7.3:

- Logistic regression on HIV. We tune the q parameter for GCE based on validation data. We use $q = 0, 0, 0.7, 0.3$ respectively for the four scenarios we consider. For RobustlyRegRisk, we use $\frac{\rho}{n} = 10$ (where n is the training sample size) and we find that the performance is not sensitive to the choice of ρ. For CVaR, the tuned α value is 0.5 when the data imbalance ratio is 1:4, and 0.1 when the imbalance ratio is 1:20. For focal loss, we tune the hyperparameters for best performance and select $\gamma = 2$, α =0.5, 0.1, 0.5, and 0.2 for four scenarios. For HAR, we tune the regularization parameter λ via grid search from $\{0.1, 1, 2, 5, 10\}$ and select the best one. We use $t = -2$ for TERM in the presence of noise, and tune the positive t's based on validation data. In particular, the values of tilts under four cases are: (0, 0.1), (0, 50), (-2, 5), and (-2, 10) for TERM_{sc} and (0.1, 0), (50, 0), (1, -2) and (50, -2) for TERM_{ca}.
- ResNet32 on CIFAR10. We reproduce (and then directly take) the results from [45] for all baseline methods. For hierarchical TERM, we scale t from 0 to 3 for group-level tilting, and scale t from 0 to -2 for sample-level tilting within each group. λ is set to 0.2. We use the default hyperparameters (batch size, learning rate, etc) in the open-sourced code of HAR [45] for TERM.

Bibliography

[1] Tensorflow federated: Machine learning on decentralized data. URL `https://www.tensorflow.org/federated`.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. K. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation*, 2016.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. K. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation*, 2016.

[4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Conference on Computer and Communications Security*, 2016.

[5] S. Abdelkarim, P. Achlioptas, J. Huang, B. Li, K. Church, and M. Elhoseiny. Long-tail visual relationship recognition with a visiolinguistic hubless loss. *arXiv preprint arXiv:2004.00436*, 2020.

[6] J. Abernethy, P. Awasthi, M. Kleindessner, J. Morgenstern, C. Russell, and J. Zhang. Active sampling for min-max fairness. In *International Conference on Machine Learning*, 2022.

[7] A. Agarwal, A. Beygelzimer, M. Dudik, J. Langford, and H. Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, 2018.

[8] A. Agarwal, J. Langford, and C.-Y. Wei. Federated residual learning. *arXiv preprint arXiv:2003.12880*, 2020.

[9] A. Ahmadi-Javid. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 2012.

[10] A. Alacaoglu, Y. Malitsky, P. Mertikopoulos, and V. Cevher. A new regret analysis for adam-type algorithms. In *International Conference on Machine Learning*, 2020.

[11] M. Alaggan, S. Gambs, and A.-M. Kermarrec. Heterogeneous differential privacy. *arXiv preprint arXiv:1504.06998*, 2015.

[12] N. Aldaghri, H. Mahdavifar, and A. Beirami. Feo2: Federated learning with opt-out differential privacy. *arXiv preprint arXiv:2110.15252*, 2021.

[13] W. Alghamdi, S. Asoodeh, H. Wang, F. P. Calmon, D. Wei, and K. N. Ramamurthy. Model projection: Theory and applications to fair machine learning. In *IEEE International Symposium on Information Theory*, 2020.

[14] Z. Allen-Zhu. How to make the gradients small stochastically: Even faster convex and nonconvex sgd. In *Advances in Neural Information Processing Systems*, 2018.

[15] E. Amid, A. Ganesh, R. Mathews, S. Ramaswamy, S. Song, T. Steinke, V. M. Suriyakumar, O. Thakkar, and A. Thakurta. Public data-assisted mirror descent for private model training. *arXiv preprint arXiv:2112.00193*, 2021.

[16] G. Andrew, O. Thakkar, H. B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. In *Advances in Neural Information Processing Systems*, 2021.

[17] E. Arıkan. An inequality on guessing and its application to sequential decoding. *IEEE Transactions on Information Theory*, 1996.

[18] P. Artzner. Thinking coherently. *Risk*, 1997.

[19] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 1999.

[20] H. Asi, J. Duchi, A. Fallah, O. Javidbakht, and K. Talwar. Private adaptive gradient methods for convex optimization. In *International Conference on Machine Learning*, 2021.

[21] T. T. F. Authors. TensorFlow Federated Stack Overflow dataset, 2019. URL `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data`.

[22] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[23] S. Baharlouei, M. Nouiehed, A. Beirami, and M. Razaviyayn. Rényi fair inference. In *International Conference on Learning Representations*, 2020.

[24] R. Balakrishnan, T. Li, T. Zhou, N. Himayat, V. Smith, and J. Bilmes. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2022.

[25] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Symposium on Foundations of Computer Science*, 2014.

[26] A. Beirami, R. Calderbank, M. M. Christiansen, K. R. Duffy, and M. Médard. A characterization of guesswork on swiftly tilting curves. *IEEE Transactions on Information Theory*, 2018.

[27] G. Bennett. Probability inequalities for the sum of independent random variables.

Journal of the American Statistical Association, 1962.

[28] D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 1997.

[29] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, 2019.

[30] K. Bhatia, P. Jain, and P. Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, 2015.

[31] K. Bhatia, P. Jain, P. Kamalaruban, and P. Kar. Consistent robust regression. In *Advances in Neural Information Processing Systems*, 2017.

[32] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, 2011.

[33] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

[34] C. L. Blake. Uci repository of machine learning databases. *http://www.ics.uci.edu/~mlearn/MLRepository*, 1998.

[35] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

[36] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselande. Towards federated learning at scale: System design. In *Conference on Machine Learning and Systems*, 2019.

[37] V. S. Borkar. Q-learning for risk-sensitive control. *Mathematics of Operations Research*, 2002.

[38] S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. 2013.

[39] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2010.

[40] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[41] R. W. Butler. *Saddlepoint approximations with applications*. Cambridge University Press, 2007.

[42] G. C. Calafiore and L. El Ghaoui. *Optimization Models*. Cambridge University Press, 2014.

[43] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. In *NeurIPS Workshop on Federated*

Learning for Data Privacy and Confidentiality, 2019.

[44] F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, 2017.

[45] K. Cao, Y. Chen, J. Lu, N. Arechiga, A. Gaidon, and T. Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. In *International Conference on Learning Representations*, 2021.

[46] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Transactions on Information and System Security*, 2011.

[47] H. Chang, T. D. Nguyen, S. K. Murakonda, E. Kazemi, and R. Shokri. On adversarial bias and the robustness of fair machine learning. *arXiv preprint arXiv:2006.08669*, 2020.

[48] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, 2017.

[49] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 2011.

[50] F. Chen, Z. Dong, Z. Li, and X. He. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*, 2018.

[51] R. Chen and I. C. Paschalidis. Distributionally robust learning. *Foundations and Trends® in Optimization*, 2020.

[52] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[53] Y. J. Cho, D. Jhunjhunwala, T. Li, V. Smith, and G. Joshi. Maximizing global model appeal in federated learning. *arXiv preprint arXiv:2205.14840*, 2022.

[54] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[55] N. Cohen and A. Shashua. Simnets: A generalization of convolutional networks. *arXiv preprint arXiv:1410.0781*, 2014.

[56] N. Cohen, O. Sharir, and A. Shashua. Deep simnets. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[57] L. Collins, A. Mokhtari, and S. Shakkottai. Task-robust model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 2020.

[58] A. Cotter, H. Jiang, M. R. Gupta, S. Wang, T. Narayan, S. You, and K. Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 2019.

[59] A. Cotter, H. Jiang, S. Wang, T. Narayan, M. Gupta, S. You, and K. Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 2019.

[60] T. M. Cover and J. A. Thomas. Information theory and statistics. *Elements of Information Theory*, 1991.

[61] S. Curi, K. Y. Levy, S. Jegelka, and A. Krause. Adaptive sampling for stochastic risk-averse learning. In *Advances in Neural Information Processing Systems*, 2020.

[62] M. Dashti, P. Azmi, and K. Navaie. Harmonic mean rate fairness for cognitive radio networks with heterogeneous traffic. *Transactions on Emerging Telecommunications Technologies*, 2013.

[63] S. De, A. Mukherjee, and E. Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

[64] 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.

[65] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, 2012.

[66] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal Distributed Online Prediction Using Mini-Batches. *Journal of Machine Learning Research*, 2012.

[67] A. Dembo and O. Zeitouni. *Large deviations techniques and applications*. Springer Science & Business Media, 2009.

[68] Y. Deng, M. M. Kamani, and M. Mahdavi. Distributionally robust federated averaging. *Advances in Neural Information Processing Systems*, 2020.

[69] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive personalized federated learning, 2021. URL `https://openreview.net/forum?id=g0a-XYjpQ7r`.

[70] S. Denisov, B. McMahan, K. Rush, A. Smith, and A. Thakurta. Improved differential privacy for sgd via optimal private linear operators on adaptive streams. In *Advances in Neural Information Processing Systems*, 2022.

[71] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, 2019.

[72] C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, 2020.

[73] M. Donini, L. Oneto, S. Ben-David, J. S. Shawe-Taylor, and M. Pontil. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, 2018.

[74] D. Dua and C. Graff. UCI machine learning repository [http://archive. ics. uci. edu/ml]. https://archive. ics. uci. edu/ml/datasets. 2019.

[75] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 2004.

[76] J. Duchi and H. Namkoong. Learning models with uniform performance via distributionally robust optimization. *arXiv preprint arXiv:1810.08750*, 2018.

[77] J. Duchi and H. Namkoong. Variance-based regularization with convex objectives. *Journal of Machine Learning Research*, 2019.

[78] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

[79] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Privacy aware learning. In *Advances in Neural Information Processing Systems*, 2012.

[80] J. Dumford and W. Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. *arXiv preprint arXiv:1812.03128*, 2018.

[81] P. Dupuis and R. S. Ellis. *A Weak Convergence Approach to the Theory of Large Deviations*. 1997.

[82] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, 2006.

[83] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science*, 2012.

[84] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 2014.

[85] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *International Conference on Embedded Networked Sensor Systems*, 2004.

[86] A. Eryilmaz and R. Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications*, 2006.

[87] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *International Conference on Knowledge Discovery and Data Mining*, 2004.

[88] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems*, 2020.

[89] Y. Fan, S. Lyu, Y. Ying, and B. Hu. Learning with average top-k loss. In *Advances in Neural Information Processing Systems*, 2017.

[90] M. Fang, X. Cao, J. Jia, and N. Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[91] M. Feldman. Computational fairness: Preventing machine-learned discrimination. 2015.

[92] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[93] H. Föllmer and T. Knispel. Entropic risk measures: Coherence vs. convexity, model ambiguity and robust large deviations. *Stochastics and Dynamics*, 2011.

[94] H. Föllmer and A. Schied. Stochastic finance: an introduction in discrete time.

2004.

[95] R. G. Gallager. *Information theory and reliable communication*. Springer, 1968.

[96] J. Gao, H. Jagadish, and B. C. Ooi. Active sampler: Light-weight accelerator for complex data analytics at scale. *arXiv preprint arXiv:1512.03880*, 2015.

[97] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, 2015.

[98] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013.

[99] S. Ghadimi, A. Ruszczynski, and M. Wang. A single timescale stochastic approximation method for nested stochastic optimization. *SIAM Journal on Optimization*, 2020.

[100] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. An efficient framework for clustered federated learning. In *Advances in Neural Information Processing Systems*, 2020.

[101] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 2009.

[102] M. Goldblum, S. Reich, L. Fowl, R. Ni, V. Cherepanova, and T. Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. *arXiv preprint arXiv:2002.06753*, 2020.

[103] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[104] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

[105] M. Gürbüzbalaban, A. Ruszczyński, and L. Zhu. A stochastic subgradient method for distributionally robust non-convex and non-smooth learning. *Journal of Optimization Theory and Applications*, 2022.

[106] E. L. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal on Selected Areas in communications*, 1991.

[107] F. Hanzely and P. Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[108] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik. Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 2020.

[109] W. Hao, N. Mehta, K. J. Liang, P. Cheng, M. El-Khamy, and L. Carin. Waffle: Weight anonymized factorization for federated learning. *arXiv preprint arXiv:2008.05687*, 2020.

[110] Y. Hao, J. Rong, and Y. Sen. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, 2019.

[111] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

[112] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 2015.

[113] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, 2018.

[114] L. He, S. P. Karimireddy, and M. Jaggi. Byzantine-robust learning on heterogeneous datasets via resampling. In *NeurIPS Workshop on Scalability, Privacy, and Security in Federated Learning*, 2020.

[115] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in Neural Information Processing Systems*, 2018.

[116] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

[117] G. Hinton, N. Srivastava, and K. Swersky. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning, Coursera Lecture 6e*, 2012.

[118] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*. 1994.

[119] M. Holland and K. Ikeda. Better generalization with less data using robust gradient descent. In *International Conference on Machine Learning*, 2019.

[120] R. A. Howard and J. E. Matheson. Risk-sensitive markov decision processes. *Management science*, 1972.

[121] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu. FedMGDA+: Federated learning meets multi-objective optimization. *arXiv preprint arXiv:2006.11489*, 2020.

[122] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An in-depth study of lte: effect of network protocol and application behavior on performance. *SIGCOMM Computer Communication Review*, 2013.

[123] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu. Loadaboost: Loss-based adaboost federated machine learning on medical data. *arXiv preprint arXiv:1811.12629*, 2018.

[124] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. In *Advances in Neural Information Processing Systems*, 2020.

[125] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 1964.

[126] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation*, 1984.

[127] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[128] A. H. Jiang, D. L.-K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

[129] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[130] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2018.

[131] P. Jiang and G. Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, 2018.

[132] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[133] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, 2017.

[134] C. Jin, P. Netrapalli, and M. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International Conference on Machine Learning*, 2020.

[135] I. Jindal, M. Nokleby, and X. Chen. Learning deep networks from noisy labels with dropout regularization. In *International Conference on Data Mining*, 2016.

[136] I. Jindal, D. Pressel, B. Lester, and M. Nokleby. An effective label noise model for dnn text classification. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[137] Z. Jorgensen, T. Yu, and G. Cormode. Conservative or liberal? personalized differential privacy. In *International Conference on Data Engineering*, 2015.

[138] P. Jorion. Value at risk: a new benchmark for measuring derivatives risk. *Irwin Professional Pub*, 1996.

[139] S. Kaczmarz. Approximate solution of systems of linear equations. *International Journal of Control*, 1993.

[140] Kaggle. Stack Overflow Data on Kaggle. `https://www.kaggle.com/datasets/stackoverflow/stackoverflow`, 2022.

[141] P. Kairouz, M. R. Diaz, K. Rush, and A. Thakurta. (nearly) dimension independent private erm with adagrad rates via publicly estimated subspaces. In *Conference on Learning Theory*, 2021.

[142] P. Kairouz, M. R. Diaz, K. Rush, and A. Thakurta. (nearly) dimension independent private erm with adagrad rates via publicly estimated subspaces. In *Conference on Learning Theory*, 2021.

[143] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, 2021.

[144] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021.

[145] D. S. Kalogerias and W. B. Powell. Recursive optimization of convex risk measures: Mean-semideviation models. *arXiv preprint arXiv:1804.00636*, 2018.

[146] M. M. Kamani, F. Haddadpour, R. Forsati, and M. Mahdavi. Efficient fair principal component analysis. *arXiv preprint arXiv:1911.04931*, 2019.

[147] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2016.

[148] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 2020.

[149] A. Katharopoulos and F. Fleuret. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017.

[150] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 1997.

[151] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 1998.

[152] A. Khetan, Z. C. Lipton, and A. Anandkumar. Learning from noisy singly-labeled data. In *International Conference on Learning Representations*, 2018.

[153] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, 2019.

[154] M. Khodak, R. Tu, T. Li, L. Li, M.-F. Balcan, V. Smith, and A. Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *arXiv preprint arXiv:2106.04502*, 2021.

[155] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[156] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[157] B. W. Kort and D. P. Bertsekas. A new penalty function method for constrained minimization. In *IEEE Conference on Decision and Control and 11th Symposium on Adaptive Processes*, 1972.

[158] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[159] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 2010.

[160] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui. A superquantile approach for federated learning with heterogeneous devices. In *Annual Conference on Information Sciences and Systems*, 2021.

[161] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. Chi. Fairness without demographics through adversarially reweighted learning. *Advances in Neural Information Processing Systems*, 2020.

[162] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

[163] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*. 2019.

[164] T. Lan, D. Kao, M. Chiang, and A. Sabharwal. An axiomatic theory of fairness in network resource allocation. In *Conference on Information Communications*, 2010.

[165] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[166] L. Leqi, A. Prasad, and P. K. Ravikumar. On human-aligned risk minimization. In *Advances in Neural Information Processing Systems*, 2019.

[167] D. Levy, Y. Carmon, J. C. Duchi, and A. Sidford. Large-scale methods for distributionally robust optimization. In *Advances in Neural Information Processing Systems*, 2020.

[168] K. Y. Levy, A. Yurtsever, and V. Cevher. Online adaptive methods, universality and acceleration. *Advances in Neural Information Processing Systems*, 2018.

[169] J. Li, M. Khodak, S. Caldas, and A. Talwalkar. Differentially private meta-learning. In *International Conference on Learning Representations*, 2020.

[170] J. Li, M. Khodak, S. Caldas, and A. Talwalkar. Differentially private meta-learning. In *International Conference on Learning Representations*, 2020.

[171] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *AAAI Conference on Artificial Intelligence*, 2019.

[172] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, 2014.

[173] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic

optimization. In *Conference on Knowledge Discovery and Data Mining*, 2014.

[174] M. Li, M. Soltanolkotabi, and S. Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[175] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy. FedDANE: A federated newton-type method. In *Asilomar Conference on Signals, Systems, and Computers*, 2019.

[176] T. Li, A. K. Sahu, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *Conference on Machine Learning and Systems*, 2020.

[177] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, Special Issue on Distributed, Streaming Machine Learning*, 2020.

[178] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 2020.

[179] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *Conference on Machine Learning and Systems*, 2020.

[180] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Feddane: A federated newton-type method. *arXiv preprint arXiv:2001.01920*, 2020.

[181] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *Conference on Machine Learning and Systems*, 2020.

[182] T. Li, M. Sanjabi, A. Beirami, and V. Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020.

[183] T. Li, A. Beirami, M. Sanjabi, and V. Smith. Tilted empirical risk minimization. In *International Conference on Learning Representations*, 2021.

[184] T. Li, A. Beirami, M. Sanjabi, and V. Smith. Tilted empirical risk minimization. In *International Conference on Learning Representations*, 2021.

[185] T. Li, A. Beirami, M. Sanjabi, and V. Smith. On tilted losses in machine learning: Theory and applications. *arXiv preprint arXiv:2109.06141*, 2021.

[186] T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, 2021.

[187] T. Li, M. Zaheer, S. J. Reddi, and V. Smith. Private adaptive optimization with side information. In *International Conference on Machine Learning*, 2022.

[188] T. Li, M. Zaheer, K. Liu, S. J. Reddi, H. B. McMahan, and V. Smith. Differentially private adaptive optimization with delayed preconditioners. In *International Conference on Learning Representations*, 2023.

[189] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of fedavg

on non-iid data. In *International Conference on Learning Representations*, 2020.

[190] P. P. Liang, T. Liu, L. Ziyin, R. Salakhutdinov, and L.-P. Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[191] T. Lin, S. U. Stich, and M. Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020.

[192] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision*, 2017.

[193] G.-H. Liu and E. A. Theodorou. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv preprint arXiv:1908.10920*, 2019.

[194] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning attack on neural networks. In *Network and Distributed System Security Symposium*, 2018.

[195] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, 2015.

[196] B. London. PAC identifiability in federated personalization. In *NeurIPS 2020 Workshop on Scalability, Privacy, and Security in Federated Learning*, 2020.

[197] A. Lowy and M. Razaviyayn. Output perturbation for differentially private convex optimization with improved population loss bounds, runtimes and applications to private adversarial training. *arXiv preprint arXiv:2102.04704*, 2021.

[198] A. Lowy, S. Baharlouei, R. Pavan, M. Razaviyayn, and A. Beirami. A stochastic optimization framework for fair risk minimization. *arXiv preprint arXiv:2102.12586*, 2021.

[199] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[200] H. Mahdavifar, A. Beirami, B. Touri, and J. S. Shamma. Global games with noisy information sharing. *IEEE Transactions on Signal and Information Processing over Networks*, 2018.

[201] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *International Conference on Computer Vision*, 2011.

[202] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[203] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, 1999.

[204] J. L. Massey. Guessing and entropy. In *IEEE International Symposium on Information Theory*, 1994.

[205] A. Maurer and M. Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

[206] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas.

Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[207] H. B. McMahan and M. J. Streeter. Adaptive bound optimization for online convex optimization. In *Conference on Learning Theory*, 2010.

[208] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[209] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[210] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *International Conference on Learning Representations*, 2018.

[211] A. K. Menon, A. S. Rawat, S. J. Reddi, and S. Kumar. Can gradient clipping mitigate label noise? In *International Conference on Learning Representations*, 2020.

[212] N. Merhav. List decoding—Random coding exponents and expurgated exponents. *IEEE Transactions on Information Theory*, 2014.

[213] I. Mironov, K. Talwar, and L. Zhang. Rényi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[214] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 2000.

[215] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, 2019.

[216] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor. Fedfast: Going beyond average for faster training of federated recommender systems. In *International Conference on Knowledge Discovery & Data Mining*, 2020.

[217] B. Mukhoty, G. Gopakumar, P. Jain, and P. Kar. Globally-convergent iteratively reweighted least squares for robust regression problems. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[218] M. C. Mukkamala and M. Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International Conference on Machine Learning*, 2017.

[219] H. Namkoong and J. C. Duchi. Variance-based regularization with convex objectives. In *Advances in Neural Information Processing Systems*, 2017.

[220] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan. Achieving mac layer fairness in wireless packet networks. In *International Conference on Mobile Computing and Networking*, 2000.

[221] M. J. Neely, E. Modiano, and C.-P. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions On Networking*, 2008.

[222] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 2009.

[223] M. Nouiehed, M. Sanjabi, T. Huang, J. D. Lee, and M. Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems*, 2019.

[224] I. Olier, N. Sadawi, G. R. Bickerton, J. Vanschoren, C. Grosan, L. Soldatova, and R. D. King. Meta-qsar: a large-scale application of meta-learning to drug design and discovery. *Machine Learning*, 2018.

[225] D. M. Ostrovskii, A. Lowy, and M. Razaviyayn. Efficient search of first-order nash equilibria in nonconvex-concave smooth min-max problems. *arXiv preprint arXiv:2002.07919*, 2020.

[226] R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 1997.

[227] E. Pee and J. O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*, 2011.

[228] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, 2014.

[229] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

[230] K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[231] F. Prost, H. Qian, Q. Chen, E. H. Chi, J. Chen, and A. Beutel. Toward a better trade-off between performance and fairness with kernel-based distribution matching. *arXiv preprint arXiv:1910.11779*, 2019.

[232] Q. Qi, Z. Guo, Y. Xu, R. Jin, and T. Yang. A practical online method for distributionally deep robust optimization. *arXiv preprint arXiv:2006.10138*, 2020.

[233] Q. Qi, Y. Xu, R. Jin, W. Yin, and T. Yang. Attentional biased stochastic gradient for imbalanced classification. *arXiv preprint arXiv:2012.06951*, 2020.

[234] B. Radunovic and J.-Y. Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 2007.

[235] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[236] S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

[237] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[238] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for

robust deep learning. In *International Conference on Machine Learning*, 2018.

[239] A. Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961.

[240] A. Rezaei, R. Fathony, O. Memarrast, and B. D. Ziebart. Fairness for robust log loss classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[241] P. Richtárik and M. Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 2016.

[242] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 2002.

[243] R. T. Rockafellar, S. Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2000.

[244] Y. Roh, K. Lee, S. E. Whang, and C. Suh. Fr-train: A mutual information-based approach to fair and robust training. In *International Conference on Machine Learning*, 2020.

[245] T. Rögnvaldsson. UCI repository of machine learning databases. *https://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage*, 2013.

[246] S. Salamatian, L. Liu, A. Beirami, and M. Médard. Mismatched guesswork. *arXiv preprint arXiv:1907.00531*, 2019.

[247] S. Samadi, U. Tantipongpipat, J. H. Morgenstern, M. Singh, and S. Vempala. The price of fair PCA: One extra dimension. In *Advances in Neural Information Processing Systems*, 2018.

[248] M. Sanjabi, M. Razaviyayn, and Z.-Q. Luo. Optimal joint base station assignment and beamforming for heterogeneous networks. *IEEE Transactions on Signal Processing*, 2014.

[249] F. Sattler, K.-R. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[250] M. Schmidt and N. L. Roux. Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*, 2013.

[251] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

[252] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[253] O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International Conference on Machine Learning*, 2014.

[254] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming:*

modeling and theory. 2014.

[255] C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[256] H. Shi, R. V. Prasad, E. Onur, and I. Niemegeers. Fairness in wireless networks: Issues, measures and challenges. *IEEE Communications Surveys and Tutorials*, 2014.

[257] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[258] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, 2019.

[259] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, 2019.

[260] D. Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, 1976.

[261] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, K. Rush, and S. Prakash. Federated reconstruction: Partially local federated learning. *arXiv preprint arXiv:2102.03448*, 2021.

[262] A. Sinha, H. Namkoong, and J. Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[263] A. Słowik and L. Bottou. On distributionally robust optimization and data rebalancing. In *International Conference on Artificial Intelligence and Statistics*, 2022.

[264] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 2017.

[265] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 2017.

[266] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 2018.

[267] V. Smith, S. Forte, C. Ma, M. Takac, M. I. Jordan, and M. Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 2018.

[268] T. Soma and Y. Yoshida. Statistical learning with conditional value at risk. *arXiv preprint arXiv:2002.05826*, 2020.

[269] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *IEEE Conference on Signal and Information Processing*, 2013.

[270] S. Song, T. Steinke, O. Thakkar, and A. Thakurta. Evading the curse of dimensionality in unconstrained private glms via private gradient descent. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[271] I. Stelmakh, N. B. Shah, and A. Singh. PeerReview4All: Fair and accurate reviewer assignment in peer review. In *Algorithmic Learning Theory*, 2019.

[272] S. U. Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

[273] M. Streeter and H. B. McMahan. Less regret via online conditioning. *arXiv preprint arXiv:1002.4862*, 2010.

[274] T. Strohmer and R. Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 2009.

[275] P. Subramani, N. Vadivelu, and G. Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. In *Advances in Neural Information Processing Systems*, 2021.

[276] G. Sun, Y. Cong, J. Dong, Q. Wang, and J. Liu. Data poisoning attacks on federated machine learning. *arXiv preprint arXiv:2004.10020*, 2020.

[277] Z. Sun, P. Kairouz, A. T. Suresh, and H. McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[278] T. Sypherd, M. Diaz, J. K. Cava, G. Dasarathy, P. Kairouz, and L. Sankar. A tunable loss function for robust classification: Calibration, landscape, and generalization. *arXiv preprint arXiv:1906.02314*, 2019.

[279] A. Szabo, H. Jamali-Rad, and S.-D. Mannava. Tilted cross entropy (TCE): Promoting fairness in semantic segmentation. *arXiv preprint arXiv:2103.14051*, 2021.

[280] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[281] U. Tantipongpipat, S. Samadi, M. Singh, J. H. Morgenstern, and S. Vempala. Multi-criteria dimensionality reduction with applications to fairness. In *Advances in Neural Information Processing Systems*, 2019.

[282] C. Van Berkel. Multi-core for mobile phones. In *Conference on Design, Automation and Test in Europe*, 2009.

[283] S. Vaswani, F. Bach, and M. Schmidt. Fast and faster convergence of sgd for over-parameterized models (and an accelerated perceptron). In *International Conference on Artificial Intelligence and Statistics*, 2019.

[284] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[285] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural*

Information Processing Systems, 2018.

[286] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 2008.

[287] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 2005.

[288] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Advances in Neural Information Processing Systems*, 2020.

[289] J. Wang and G. Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

[290] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[291] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[292] M. Wang, J. Liu, and E. Fang. Accelerating stochastic composition optimization. *Advances in Neural Information Processing Systems*, 2016.

[293] M. Wang, E. X. Fang, and H. Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 2017.

[294] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 2019.

[295] X. Wang, Y. Jiang, M. Huang, and H. Zhang. Robust variable selection with exponential squared loss. *Journal of the American Statistical Association*, 2013.

[296] Z. Wang, T. Oates, and J. Lo. Adaptive normalized risk-averting training for deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2016.

[297] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 2020.

[298] H. Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 1912.

[299] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro. Learning non-discriminatory predictors. In *Conference on Learning Theory*, 2017.

[300] B. E. Woodworth, J. Wang, A. Smith, B. McMahan, and N. Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In *Advances in Neural Information Processing Systems*, 2018.

[301] S. Wu, T. Li, Z. Charles, Y. Xiao, Z. Liu, Z. Xu, and V. Smith. Motley: Bench-

marking heterogeneity and personalization in federated learning. *arXiv preprint arXiv:2206.09262*, 2022.

[302] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[303] C. Xie, K. Huang, P.-Y. Chen, and B. Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[304] X. Xu and L. Lyu. Towards building a robust and fair federated learning system. *arXiv preprint arXiv:2011.10464*, 2020.

[305] J. Yang and J. S. Rosenthal. Complexity results for mcmc derived from quantitative bounds. *arXiv preprint arXiv:1708.00829*, 2017.

[306] M. Yang, L. Xu, M. White, D. Schuurmans, and Y.-l. Yu. Relaxed clipping: A global training method for robust regression and classification. In *Advances in Neural Information Processing Systems*, 2010.

[307] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 2007.

[308] I.-C. Yeh and C.-h. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 2009.

[309] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 2018.

[310] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[311] H. Yu, S. Yang, and S. Zhu. Parallel restarted sgd for non-convex optimization with faster convergence and less communication. In *AAAI Conference on Artificial Intelligence*, 2019.

[312] T. Yu, E. Bagdasaryan, and V. Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[313] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, and S. Seshan. Learning context-aware policies from multiple smart homes via federated multi-task learning. In *International Conference on Internet-of-Things Design and Implementation*, 2020.

[314] Y.-l. Yu, Ö. Aslan, and D. Schuurmans. A polynomial-time form of robust regression. In *Advances in Neural Information Processing Systems*, 2012.

[315] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *International Conference on World Wide Web*, 2017.

[316] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *International Conference on Artificial Intelligence*

and Statistics, 2017.

[317] M. B. Zafar, I. Valera, M. Gomez-Rodriguez, and K. P. Gummadi. Fairness constraints: A flexible approach for fair classification. *Journal of Machine Learning Research*, 2019.

[318] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2018.

[319] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[320] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. Reddi, S. Kumar, and S. Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, 2020.

[321] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. J. Reddi, S. Kumar, and S. Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, 2020.

[322] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2021.

[323] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, 2015.

[324] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 2015.

[325] Y. Zhang, J. C. Duchi, and M. J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 2013.

[326] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, 2018.

[327] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[328] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[329] F. Zhou and G. Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *International Joint Conference on Artificial Intelligence*, 2018.

[330] P. Zhou, X. Yuan, H. Xu, S. Yan, and J. Feng. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, 2019.

[331] T. Zhou, S. Wang, and J. Bilmes. Robust curriculum learning: from clean label detection to noisy label self-correction. In *International Conference on Learning Representations*, 2021.

[332] Y. Zhou, X. Chen, M. Hong, Z. S. Wu, and A. Banerjee. Private stochastic non-convex optimization: Adaptive algorithms and tighter generalization bounds. *arXiv preprint arXiv:2006.13501*, 2020.

[333] Y. Zhou, Z. S. Wu, and A. Banerjee. Bypassing the ambient dimension: Private sgd with gradient subspace identification. In *International Conference on Learning Representations*, 2021.

[334] L. Zhu, M. Gürbüzbalaban, and A. Ruszczyński. Distributionally robust learning with weakly convex losses: Convergence rates and finite-sample guarantees. *arXiv preprint arXiv:2301.06619*, 2023.