Distributed Learning with Adversarial Gradient Perturbations

Nawapon Sangsiri[1] and **Yufei Tao**[1,2]
[1]The Chinese University of Hong Kong
[2]Nanyang Technological University
{sangsiri, taoyf}@cse.cuhk.edu.hk

Abstract

Privacy concerns in distributed learning often lead clients to return intentionally altered gradient information. We consider the problem of learning convex and L-smooth functions under *adversarial gradient perturbation*, where a client's gradient reply to a server query can deviate arbitrarily from the true gradient subject to a distance bound. Our study focuses on two fundamental questions: (i) what is the smallest achievable *sub-optimality gap* (i.e., excess error in optimization) under such responses, and (ii) how many queries are sufficient to guarantee a given sub-optimality gap? We establish tight feasibility thresholds on the sub-optimality gap and provide algorithms that achieve these thresholds with provable query complexity guarantees.

1 Introduction

Modern learning systems increasingly rely on edge-based architectures, where local devices compute gradient information and communicate it to a central server. In such environments, local gradient reports may be deliberately perturbed to protect sensitive data. This motivates the study of learning algorithms that operate under *adversarially corrupted gradients*, where each individual gradient reply may deviate arbitrarily from the true gradient, subject only to a known bound on the perturbation magnitude. Understanding what can and cannot be learned in this setting, as well as the query complexity required to do so, is a fundamental question.

Formally, we consider a setting where there are n geographically distributed clients, each holding a convex and L-smooth *loss function* $\ell_i : \mathbb{R}^d \to \mathbb{R}$ where $i \in [n]$.[1] Define

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w). \quad (1)$$

which is minimized at some $w^* \in \mathbb{R}^d$ — when multiple minimizers exist, we define w^* to be one with the smallest $\|w^*\|$.[2] The *sub-optimality gap* of a vector $w \in \mathbb{R}^d$ is defined as the difference $f(w) - f(w^*)$. A server's goal is to find a vector whose sub-optimality gap is bounded by $\tau > 0$.

Communication between the server and clients is limited to the following query primitive:

> **Approximate Gradient Query:** The server selects an $i \in [n]$ and sends the i-th client a vector $w \in \mathbb{R}^d$. The client replies with a vector v satisfying
>
> $$\|v - \nabla \ell_i(w)\| \leq \epsilon \quad (2)$$
>
> where ϵ is a value agreed upon in advance by the server and all clients.

Crucially, the client may respond with an *arbitrary* vector v satisfying (2), including one chosen adversarially so as to impede the server's optimization procedure. Larger values of ϵ expand the set of permissible replies and thus offer stronger privacy protection on the local data contributing to the computation of $\nabla \ell_i(w)$.

We refer to the above problem as *distributed learning with adversarial gradient perturbations* (DLAGP). Our goal is to characterize the fundamental feasibility of learning and derive interesting query bounds in this setting. In particular, we aim to answer two questions:

- **Q1**: How small can the sub-optimality gap be made? Formally, what is the smallest value τ_{min} such that, for all legal inputs $\{\ell_1, \ell_2, ..., \ell_n\}$, it is possible for the server to find a vector w satisfying $f(w) \leq f(w^*) + \tau_{min}$, possibly using infinitely many queries?
- **Q2**: Given a value τ that is not "unreasonably small", how many approximate gradient queries are needed for the server to guarantee a sub-optimality gap at most τ?

Our Results and Implications. We establish non-trivial answers to both questions above.

- Answers for **Q1** — If no upper bound on $\|w^*\|$ is known in advance, then no algorithm, even with infinitely many queries, can guarantee finding a vector w with a bounded sub-optimality gap, regardless of the target value τ.

  Motivated by this impossibility result, we consider R-*instances* of the DLAGP problem, where a bound $\|w^*\| \leq R$ is known. For this class of instances, we show that the smallest achievable sub-optimality gap τ_{min} lies in the interval $[\frac{1}{2}\epsilon R, 5\epsilon R]$. In other words,

[1] $[x]$ represents the set $\{1, 2, ..., x\}$ for an integer $x \geq 1$.
[2] All norms in this paper refer to the L_2 norm.

when $\tau < \epsilon R/2$, no algorithm can guarantee a sub-optimality gap of τ over all R-instances, even with infinitely many queries. Conversely, the server can always achieve a sub-optimality gap at most τ for all R-instances whenever $\tau \geq 5\epsilon R$.

- Answers for **Q2** — For any R-instance satisfying the condition $\tau \geq 5\epsilon R$, the server can deterministically ensure a sub-optimality gap at most τ using

$$O(n \cdot \min\{LR^2/\tau, LR/\epsilon\}) \tag{3}$$

queries. For any R-instance satisfying the condition $\tau \geq 5.01 \cdot \epsilon R$ (the constant 5.01 can be replaced with any value larger than 5), the server can, with probability at least $1 - \delta$, ensure a sub-optimality gap at most τ using

$$\tilde{O}\left(\frac{LR^4(B_0 + LR)^2}{\tau^3}\right) \tag{4}$$

queries, where

$$B_0 = \max_{i=1}^{n} \|\nabla \ell_i(\mathbf{0})\|. \tag{5}$$

and $\tilde{O}(.)$ hides factors logarithmic to $L, R, 1/\tau$, and $1/\delta$; note that (4) does not depend on n or d. All queries in this algorithm select clients uniformly at random. By standard sampling arguments, when n is sufficiently large, with high probability each client is queried at most once, and most clients are never queried.

2 Related Work

In the study of *oracle complexity* in convex optimization — an area comprehensively surveyed by [Bubeck, 2015] — the goal is to minimize a convex function $f : \mathbb{R}^d \to \mathbb{R}$ by making the fewest calls to an "oracle". Given a $w \in \mathbb{R}^d$, the oracle returns an exact or approximate derivative of f of a specific order at w. For example, an exact zeroth-order oracle returns $f(w)$, while a first-order counterpart returns $\nabla f(w)$. Our study contributes to the research on first-order oracles, of which see representative works [Nemirovsky and Yudin, 1983; Tsybakov, 2003; d'Aspremont, 2008; Nemirovski *et al.*, 2009; Agarwal *et al.*, 2012; Dekel *et al.*, 2012; Nesterov, 2012; Jaggi, 2013; Devolder *et al.*, 2014; Kohler and Lucchi, 2017; Lan, 2020; Mirzasoleiman *et al.*, 2020; Huang *et al.*, 2021; Lei and Tang, 2021; Boob *et al.*, 2023; Chai *et al.*, 2023; Garrigos and Gower, 2024; Kerger *et al.*, 2024; Hallak and Levy, 2024] and the references therein.

In the traditional studies of "inexact gradient methods", an approximate first-order oracle returns a stochastically perturbed version of $\nabla f(w)$, typically a random $\texttt{ora}(w) \in \mathbb{R}^d$ whose expectation is $\nabla f(w)$. In some cases, there may be a "concentration property" to limit how often $\texttt{ora}(w)$ can deviate from $\nabla f(w)$ significantly. The seminal work of [Devolder *et al.*, 2014] pioneered the concept of non-stochastic inexact first-order oracles with bounded errors in smooth convex optimization. Our permutation model — formalized as approximate gradient queries (see (2)) — aligns with their concept. Indeed, those queries promise neither expectation nor concentration guarantees, and their answers can be adversarial subject to bounded deviation. This model has gained attention in recent years [Chang *et al.*, 2022; Izzo *et al.*, 2022; Wang and Tan, 2024; Wang and Tan, 2025] — we will discuss the findings of these papers in Section 3.4.

A parallel line of work studies Byzantine-resilient distributed optimization; see, e.g., [Blanchard *et al.*, 2017; Alistarh *et al.*, 2018; Yin *et al.*, 2018; Karimireddy *et al.*, 2021] and the references therein. These works typically assume that most clients return exact or stochastically perturbed gradients, while a fraction of "Byzantine" clients may behave arbitrarily. Algorithms in this literature crucially depend on the existence of sufficiently many honest clients and exploit redundancy through robust aggregation schemes. In contrast, our model does not posit any well-behaved clients: every gradient query may be adversarially perturbed, and no stochastic population-level structure is available. As a result, classical Byzantine-robust techniques do not apply in our scenarios.

Loosely speaking, our work is also related to *federated learning*, which aims to train models across geographically distributed clients while allowing them to keep data localized; see [Li *et al.*, 2020] for an excellent introduction. A central focus in that area, however, is handling heterogeneous data distributions and ensuring convergence despite differences across client datasets. In contrast, our study abstracts away data heterogeneity and instead examines fundamental limits on learnability, as well as query complexity, under worst-case, bounded adversarial gradient perturbations.

3 Foundation: The Theory with One Client

This section considers DLAGP with only $n = 1$ client. As we will see Section 4, the theory developed in this section plays a vital role in solving the problem in its general form.

When $n = 1$, DLAGP can be rephrased in a more succinct manner. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function such that:

- **C1:** f is convex and L-smooth;
- **C2:** f is minimized at some $w^* \in \mathbb{R}^d$; if multiple minimizers exist, define w^* to be one with the smallest norm.

The goal of optimization is to find a vector $w \in \mathbb{R}^d$ whose sub-optimality gap $f(w) - f(w^*)$ is bounded by a given τ. The only way for a learner algorithm to glean information about f is to interact with an *adversarial gradient perturbation* (AGP) oracle. Given a $w \in \mathbb{R}^d$, the oracle returns a vector $\texttt{ora}(w) \in \mathbb{R}^d$ satisfying

$$\|\texttt{ora}(w) - \nabla f(w)\| \quad \leq \quad \epsilon \tag{6}$$

where $\epsilon \geq 0$; we will call it an *ϵ-AGP oracle*. Our goal is to understand *how well* and *fast* optimization can be done when $\texttt{ora}(w)$ can be an arbitrary vector obeying (6).

3.1 No Norm Bounds, No Call Bounds

We start with a result showing that if nothing is known about $\|w^*\|$, it is impossible to guarantee a sub-optimality gap τ with a finite number of oracle calls.

Theorem 1. *Fix any $L \geq 1$, any $\epsilon \geq 0$, and any $\tau > 0$. If no upper bound on $\|w^*\|$ is known in advance, no algorithm can guarantee — for every function $f : \mathbb{R} \to \mathbb{R}$ satisfying conditions* **C1**, **C2**, *and $\|\nabla f(w)\| \leq 3\tau$ for all $w \in \mathbb{R}$ — finding a $w \in \mathbb{R}$ with $f(w) \leq f(w^*) + \tau$ by calling an ϵ-AGP oracle a finite number of times.*

Proof. It suffices to prove the theorem for $\epsilon = 0$ (if $\epsilon > 0$, simply treat a 0-AGP oracle as an ϵ-oracle and apply the argument below). Let us define a function from $\mathbb{R}$ to $\mathbb{R}$,

$$f(w) = \begin{cases} 0 & \text{if } w < -R \\ (L/2)(w+R)^2 & \text{if } -R \le w \le -R + 3\tau/L \\ 3\tau w + 3\tau R - (3\tau)^2/(2L) & \text{if } w > -R + 3\tau/L \end{cases}$$

where $R > 0$ is a real value to be decided later.

As shown in the full version [Sangsiri and Tao, 2026], the function is convex, L-smooth, and ascending. Furthermore, it has the following properties:

- Property 1: $\min_w f(w) = 0$, and $f(w) = 0$ if and only if $w \le -R$.
- Property 2: $\|\nabla f(w)\| \le 3\tau$ for all $w \in \mathbb{R}$, and $\nabla f(w) = 3\tau$ for all $w > -R + 3\tau/L$.
- Property 3: $f(w) \ge 3\tau$ for any $w \ge -R + (3\tau/L) + 1$.

Let $\mathcal{A}$ be an algorithm that calls a 0-AGP oracle a finite number of calls. We will show that an adversary can force $\mathcal{A}$ to incur a sub-optimality gap larger than τ. For this purpose, the adversary initially leaves open the value of R and decides this value only after $\mathcal{A}$ has terminated.

Specifically, for any value (i.e., a 1D vector) w that $\mathcal{A}$ inquires the oracle with, the adversary always returns 3τ. Let w_i be the value that $\mathcal{A}$ provides in its i-th call to the oracle, for $i \in [t]$ where t is the (finite) number of calls. Let w_{out} be the value output by $\mathcal{A}$. Define

$$v = \min\{w_1, w_2, ..., w_t, w_{out}\}.$$

The adversary then sets

$$R = \max\{0, (3\tau/L) - v + 1\}$$

which ensures $v \ge -R + (3\tau/L) + 1$.

The reader can verify that the function f — which is now finalized — has $\nabla f(w) = 3\tau$ at every $w \in \{w_1, w_2, ..., w_t\}$. In other words, the oracle's answers to $\mathcal{A}$ have been correct. Nevertheless, $f(w_{out}) \ge f(v) \ge 3\tau$. Hence, $\mathcal{A}$ has failed to ensure $f(w_{out}) \le f(w^*) + \tau = \tau$.

The above argument works only when the adversary does not promise any upper bound on $\|w^*\|$ in advance. Indeed, $\|w^*\|$ equals R, which can be arbitrarily big because it depends on the value of v. □

3.2 Sub-Optimality Gap

Define

$$\mathcal{F}(R) = \{f \mid f \text{ satisfies } \mathbf{C1} \text{ and } \mathbf{C2} \text{ with } \|w^*\| \le R\}. \quad (7)$$

Motivated by Theorem 1, we will study functions from $\mathcal{F}(R)$ with a *known* value of R. Even in this context, learning is still impossible if the target sub-optimality gap is too small:

Theorem 2. *Fix any $R \ge 1$, any $L \ge 1$, and any $0 < \epsilon \le 1$. No algorithm working with an ϵ-AGP oracle can guarantee — for every $f \in \mathcal{F}(R)$ (recall that the definition of $\mathcal{F}(R)$ relies on L due to* **C2***) — returning a $w \in \mathbb{R}^d$ such that $f(w) < f(w^*) + \epsilon R/2$. This holds true even if the algorithm calls the oracle an infinite number of times.*

Proof. Let us define two functions from $\mathbb{R}$ to $\mathbb{R}$:

$$f_1(w) = \begin{cases} 0 & \text{if } w < -R \\ (L/2)(w+R)^2 & \text{if } -R \le w \le -R + \epsilon/L \\ \epsilon w + \epsilon R - \epsilon^2/(2L) & \text{if } w > -R + \epsilon/L \end{cases} \quad (8)$$

$$f_2(w) = f_1(-w). \quad (9)$$

As shown in the full version [Sangsiri and Tao, 2026]:

- f_1 and f_2 are convex and L-smooth;
- f_1 is ascending and f_2 is descending;
- $\|\nabla f_1(w)\| \le \epsilon$ and $\|\nabla f_2(w)\| \le \epsilon$ for all $w \in \mathbb{R}$.

Furthermore, the facts below are easy to verify:

- For f_1, the optimal vector with the smallest norm is $w_1^* = -R$ with $f_1(w_1^*) = 0$, while for f_2, the optimal vector with the smallest norm is $w_2^* = R$ with $f_2(w_2^*) = 0$. Hence, both f_1 and f_2 belong to $\mathcal{F}(R)$.
- For $w \ge 0$, $f_1(w) \ge f_1(0) = \epsilon R - \epsilon^2/(2L) \ge \epsilon R/2$.
- For $w \le 0$, $f_2(w) \ge f_2(0) = \epsilon R - \epsilon^2/(2L) \ge \epsilon R/2$.

Let $\mathcal{A}$ be an algorithm relying on an AGP oracle to minimize f. We observe that $\mathcal{A}$ cannot always distinguish between $f = f_1$ and $f = f_2$. Indeed, regardless of $f \in \{f_1, f_2\}$, when $\mathcal{A}$ solicits $\nabla f(w)$ from the oracle, the oracle can always return the value $\mathbf{0}$, which obeys (6) because $\|\nabla f(w)\| \le \epsilon$ for both $f = f_1$ and $f = f_2$.

Consider an adversary that is committed to choosing $f \in \{f_1, f_2\}$ but does so only *after* $\mathcal{A}$ has terminated. Whenever $\mathcal{A}$ calls an oracle to inquire $\nabla f(w)$, the adversary instructs the oracle to return $\mathbf{0}$, which is correct no matter $f = f_1$ or f_2. Suppose that $\mathcal{A}$ finishes by outputting a vector w_{out}. The adversary sets $f = f_2$ if $w_{out} \le 0$, or $f = f_1$ otherwise. This makes sure $f(w_{out}) \ge \epsilon R/2$, forcing the sub-optimality gap to be at least $\epsilon R/2$. □

3.3 An Optimization Algorithm

Next, we consider $f \in \mathcal{F}(R)$ and values of τ that are reasonably large. Algorithm 1 describes our optimization algorithm named `AGP-opt`.

Theorem 3. *Let f be a function from $\mathcal{F}(R)$. For any real value $\tau \ge 5\epsilon\|w^*\|$, algorithm* `AGP-opt` *finds a $w \in \mathbb{R}^d$ satisfying $f(w) \le f(w^*) + \tau$ with parameter*

$$K = \min\{5LR^2/(4\tau), LR/(4\epsilon)\}. \quad (10)$$

Furthermore, every w_k in Line 3 of `AGP-opt` *has norm at most $(17/8)R$.*

Several remarks are in order before delving into the proof:

- Gradient descent (GD) can be regarded as a special instance of `AGP-opt` with $\epsilon = 0$. The early termination condition at Line 4 is needed to guarantee a sub-optimality gap sensitive to $\|w^*\|$. The sub-optimality gap would become $O(\epsilon R)$ without early termination, as analyzed in the full version [Sangsiri and Tao, 2026].

Algorithm 1 AGP-opt(K)

1: $w_0 \leftarrow \mathbf{0}$, $k \leftarrow 0$
2: **while** $k < K$ **do**
3: $g_k \leftarrow \texttt{ora}(w_k)$ /* output of the ϵ-AGP oracle */
4: **if** $\|g_k\| < 4\epsilon$ **then return** w_k
5: $w_{k+1} \leftarrow w_k - \frac{1}{2L} g_k$ and $k \leftarrow k+1$
6: **end while**
7: **return** w_k

- Theorem 3 extends the convergence bound of GD in a seamless manner: the first term of (10) captures the influence of optimization accuracy, while the second reflects the noise imposed by the ϵ-AGP oracle. For $\epsilon = 0$, the second term disappears, leaving $K = 5LR^2/(4\tau)$, which asymptotically matches the convergence bound of GD [Garrigos and Gower, 2024, Corollary 3.5].
- Setting $w_0 = \mathbf{0}$ at Line 1 is needed for the claim (in Theorem 3) that $\|w_k\| \leq (17/8)R$ for all k. The claim will be useful in Section 4.2. Our analysis can be easily adapted to other choices of w_0 by shifting the origin of the coordinate system to w_0.

Proof of Theorem 3. Define

$$K' = \text{the value of } k \text{ when } \texttt{AGP-opt} \text{ terminates.} \tag{11}$$

Note that K' may be less than K — this happens when the termination is at Line 4. The following technical lemma is proved in the full version [Sangsiri and Tao, 2026].

Lemma 4. *Let f be a function satisfying* **C1** *and* **C2**. *Algorithm* `AGP-opt` *outputs a vector $w \in \mathbb{R}^d$ such that $f(w) - f(w^*)$ is bounded by*

$$\epsilon\|w^*\| + \max\left\{4\epsilon\|w^*\|, \frac{L \cdot \|w^*\|^2}{K}\right\}. \tag{12}$$

Furthermore, For any $k \in [0, K'-1]$, it holds that

$$\|w_{k+1} - w^*\| \leq \|w_k - w^*\| + \epsilon/(2L). \tag{13}$$

As $\tau \geq 5\epsilon\|w^*\|$ (a condition in Theorem 3), the value of (12) is at most τ when

$$K \geq \frac{L\|w^*\|^2}{\tau - \epsilon\|w^*\|}. \tag{14}$$

The inequality $\tau \geq 5\epsilon\|w^*\|$ implies $\tau - \epsilon\|w^*\| \geq (4/5)\tau$. Hence,

$$\frac{L\|w^*\|^2}{\tau - \epsilon\|w^*\|} \leq \frac{5L\|w^*\|^2}{4\tau}. \tag{15}$$

We can bound from above the right hand side of (15) in two different ways:

- As $\tau \geq 5\epsilon\|w^*\|$ gives $\|w^*\|/\tau \leq 1/(5\epsilon)$, we have
$$(15) \leq \frac{L\|w^*\|}{4\epsilon} \leq \frac{LR}{4\epsilon}$$
where the second inequality used $\|w^*\| \leq R$ (because $f \in \mathcal{F}(R)$)
- On the other hand, from $\|w^*\| \leq R$, we directly have
$$(15) \leq \frac{5LR^2}{4\tau}.$$

It now follows from (14) and Lemma 4 that the choice of K in (10) allows `AGP-opt` to output a $w \in \mathbb{R}^d$ whose sub-optimality gap is at most τ.

For each $k \in [0, K']$, Lemma 4 tells us

$$\begin{aligned} \|w_k - w^*\| &\leq \|w_0 - w^*\| + k\epsilon/(2L) \\ &= \|w^*\| + k\epsilon/(2L). \end{aligned} \tag{16}$$

For the value of K in (10), it holds that $k \leq K' \leq K \leq LR/(4\epsilon)$. Thus, $k\epsilon/(2L) \leq R/8$, from which (16) yields $\|w_k - w^*\| \leq \|w^*\| + R/8$, leading to

$$\|w_k\| \leq 2\|w^*\| + R/8 \leq (17/8)R.$$

This completes the proof of Theorem 3.

3.4 Discussion

Optimization under an ϵ-AGP oracle has been studied in [Chang *et al.*, 2022; Izzo *et al.*, 2022; Wang and Tan, 2024; Wang and Tan, 2025]. We will focus on [Izzo *et al.*, 2022; Wang and Tan, 2024] because the other two papers assume function classes different from ours.

As detailed in the full version [Sangsiri and Tao, 2026], the results of [Izzo *et al.*, 2022] could be used to show that a slightly modified version of GD finds a vector whose sub-optimality gap is

$$O\Big(\|w^*\| \cdot \sqrt{(LU/K) + B\epsilon}\Big) \tag{17}$$

where U is an upper bound on $f(w)$ for every vector w produced at any iteration of GD, and B is an upper bound on $\|\nabla f(w)\|$ over those vectors (see the full version [Sangsiri and Tao, 2026] for the precise definitions of U and B). On the other hand, by Lemma 4, we can find a vector whose sub-optimality gap is

$$O\Big(\|w^*\| \cdot \max\{\epsilon, L\|w^*\|/K\}\Big). \tag{18}$$

Our bound (18) converges faster than (17) (specifically: $1/K$ vs. $1/\sqrt{K}$). As $K \to \infty$, our bound converges to $O(\|w^*\| \cdot \epsilon)$, while (17) to $O(\|w^*\| \cdot \sqrt{B\epsilon})$. Crucially, in the context of [Izzo *et al.*, 2022], a reasonable ϵ *must* be less than B — otherwise, the fact $\|\nabla f(w)\| \leq B$ allows an ϵ-AGP oracle to *always* return vector $\mathbf{0}$, thereby giving no information to the algorithm at all. In fact, effective optimization demands $\epsilon \ll B$, in which case our sub-optimality gap $O(\|w^*\| \cdot \epsilon)$ is considerably smaller than that of [Izzo *et al.*, 2022].

[Wang and Tan, 2024] presented a mirror-descent algorithm that works when w comes from a domain with a finite radius R. In that scenario, their algorithm ensures a sub-optimality gap of

$$O\left(\frac{\text{poly}(R, L)}{\sqrt{K}} + R\epsilon\right). \tag{19}$$

The bound converges slower than (18) (K vs. $1/\sqrt{K}$). As $K \to \infty$, (19) converges to $O(R\epsilon)$ — note that the bound remains so even if $\|w^*\| \ll R$; in that scenario our bound (18), which converges to $O(\|w^*\|\epsilon)$, is significantly smaller.

4 DLAGP

We will deploy our theory in Section 3 to tackle the DLAGP problem in its general form (with an arbitrary number n of clients). Every mention of "query" henceforth refers to an approximate gradient query.

4.1 Answers for Q1

Recall that w^* is an optimal vector of f (i.e., w^* minimizes the function f defined in (1)) with the smallest norm. If no upper bound is known about $\|w^*\|$, no DLAGP algorithm can guarantee a sub-optimality gap at most τ using a finite number of queries. Indeed, any such algorithm must work in the special case of $n = 1$, which, however, gives a contradiction against Theorem 1. This is true even if the values L, ϵ, and τ are all known to the algorithm.

Next, we consider $\|w^*\| \leq R$. Recall (from our statement of **Q1** in Section 1) that τ_{min} is the minimum sub-optimality gap that we must tolerate. Theorem 2 implies

$$\tau_{min} \geq \epsilon R/2. \tag{20}$$

To see why, assume the existence of an DLAGP algorithm that can always ensure a sub-optimality gap at most τ for some $\tau < \epsilon R/2$. Such an algorithm must work in the special case of $n = 1$, which, however, gives a contradiction against Theorem 2. This is true even if the values ϵ, τ, R, and L are all known to the algorithm.

Next, we argue that $\tau_{min} \leq 5\epsilon R$, i.e., DLAGP is solvable on any R-instance as long as $\tau \geq 5\epsilon R$. As ℓ_i is L-smooth for each $i \in [n]$, so is the function f in (1). The server only needs to execute the algorithm `AGP-opt` in Section 3.3. To execute Line 3 of `AGP-opt`, which demands $\texttt{ora}(w_k)$, the server issues a query to every client $i \in [n]$, solicits the client's reply v_i, and then defines

$$\texttt{ora}(w_k) = \frac{1}{n}\sum_{i=1}^{n} v_i.$$

Accordingly,

$$\begin{aligned}\|\texttt{ora}(w_k) - \nabla f(w_k)\| &= \frac{1}{n}\Big\|\sum_{i=1}^{n}(v_i - \nabla \ell_i(w_k))\Big\| \\ &\leq \frac{1}{n}\sum_{i=1}^{n}\|v_i - \nabla \ell_i(w_k)\|\end{aligned}$$

which is at most ϵ because $\|v_i - \nabla \ell_i(w_k)\| \leq \epsilon$ for each $i \in [n]$; see (2). Theorem 3 thus permits the server to ensure a sub-optimality gap at most τ by issuing $K \cdot n$ queries, where K is given in (10). This yields the bound claimed in (3).

4.2 Answer for Q2

Next, we improve the query bound from the previous subsection when a small failure probability δ is permitted.

Center Estimation. We first take de-tour to discuss a relevant problem. Denote by $P = \{p_1, p_2, ..., p_n\}$ a set of n vectors in $\mathbb{R}^d$ such that $\|p_i\| \leq B$ for every $i \in [n]$. Let $(s_1, s_2, ..., s_m)$ be a sequence uniformly sampled from $[n]^m$. Equivalently, each s_j ($j \in [m]$) is independently drawn from $[n]$ uniformly at random. Variations of the following result have appeared before (e.g., Lemma 18 of [Kohler and Lucchi, 2017]); a proof of our version can be found in the full version [Sangsiri and Tao, 2026].

Lemma 5. *For any $t \geq 0$, by setting $m = O((B/t)^2 \log(1/\delta))$, we can guarantee*

$$\mathbf{Pr}\Big[\Big\|\frac{1}{n}\sum_{i=1}^{n} p_i - \frac{1}{m}\sum_{j=1}^{m} p_{s_j}\Big\| > t\Big] \leq \delta.$$

A Randomized Algorithm for DLAGP. Let us return to the DLAGP problem, assuming

$$\tau \geq 5.01 \cdot \epsilon R. \tag{21}$$

Suppose that Line 3 of `AGP-opt` asks for $\texttt{ora}(w_k)$. Instead of querying all clients (as done in Section 4.1), we draw m numbers $s_1, s_2, ..., s_m$ independently from $[n]$ uniformly at random — effectively sampling m clients with replacement. For each $j \in [m]$, the server issues a query to solicit a vector v_{s_j} from client s_j, which comes with the guarantee $\|v_{s_j} - \nabla \ell_{s_j}(w_k)\| \leq \epsilon$. Then, it computes

$$\texttt{ora}(w_k) = \frac{1}{m}\sum_{j=1}^{m} v_{s_j}. \tag{22}$$

To decide m, we need:

Lemma 6. *Suppose that we run `AGP-opt` by setting K as in (10). For any $i \in [n]$ and $k \in [0, K]$, it holds that $\|\nabla \ell_i(w_k)\| \leq B_0 + (17/8)LR$ (see (5) for B_0).*

The proof can be found in the full version [Sangsiri and Tao, 2026]. From now, we fix $B = B_0 + (17/8)LR$. By Lemma 5, for $m = O((\frac{B}{t})^2 \log \frac{1}{\delta'})$ — where t and δ' will be decided later — it holds that

$$\mathbf{Pr}\Big[\Big\|\frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w_k) - \frac{1}{m}\sum_{j=1}^{m} \nabla \ell_{s_j}(w_k)\Big\| > t\Big] \leq \delta'. \tag{23}$$

Note that $\frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w_k)$ is exactly $\nabla f(w_k)$. On the other hand, $\|\texttt{ora}(w_k) - \frac{1}{m}\sum_{j=1}^{m} \nabla \ell_{s_j}(w_k)\|$, which by (22) is $\frac{1}{m}\|\sum_{j=1}^{m} v_{s_j} - \sum_{j=1}^{m} \nabla \ell_{s_j}(w_k)\|$, is no more than ϵ because $\|v_{s_j} - \nabla \ell_{s_j}(w_k)\| \leq \epsilon$ for all $j \in [m]$. Plugging these into (23) gives:

$$\mathbf{Pr}\Big[\Big\|\nabla f(w_k) - \texttt{ora}(w_k)\Big\| > t + \epsilon\Big] \leq \delta'. \tag{24}$$

In other words, (22) serves the purpose of a $(t + \epsilon)$-AGP. To apply Theorem 3 and Lemma 6, we require

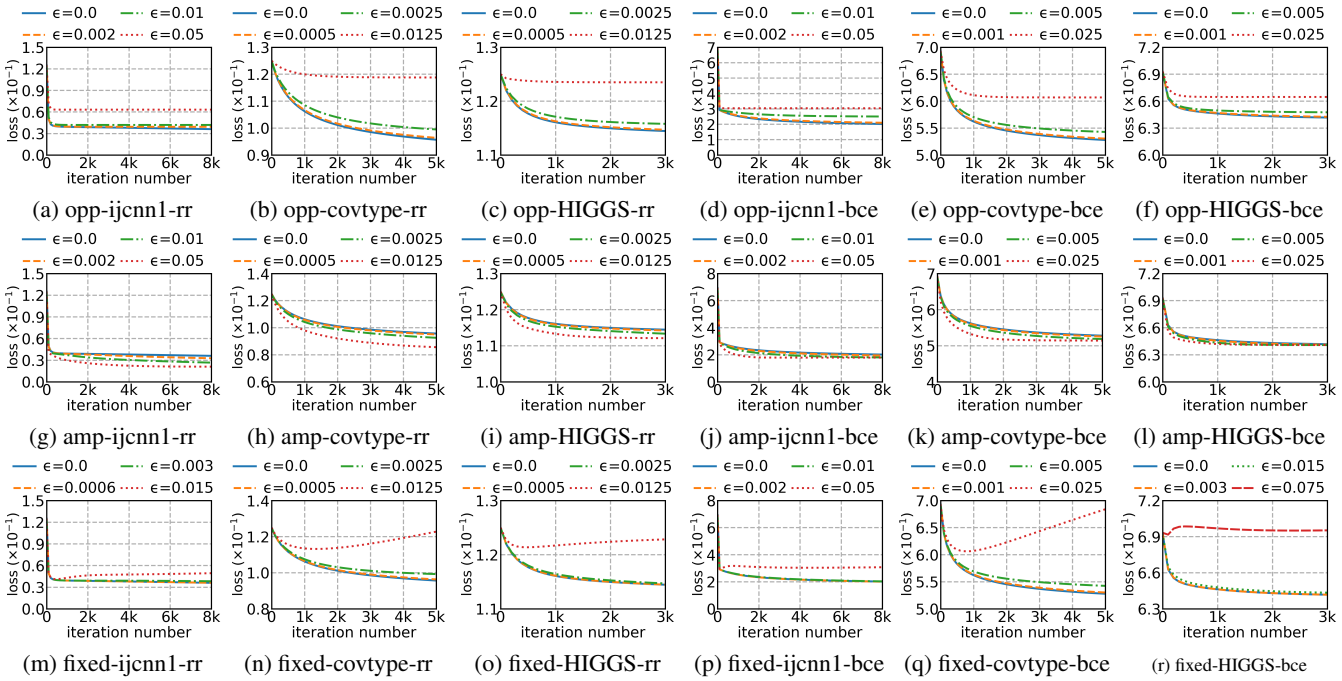
$$\tau \geq 5(t + \epsilon)R$$

as $\|w^*\| \leq R$. We therefore set

$$t = \tau/(5R) - \epsilon = \Omega(\tau/R) \tag{25}$$

where the last step used (21).

It remains to fix the value of δ'. Henceforth, set $K = 5LR^2/(4\tau)$; by Theorem 3, `AGP-opt` performs at most K iterations. By setting δ' to δ/K, we make sure that the inequality $\|\nabla f(w_k) - \texttt{ora}(w_k)\| \leq t + \epsilon$ holds for all relevant



(a) opp-ijcnn1-rr (b) opp-covtype-rr (c) opp-HIGGS-rr (d) opp-ijcnn1-bce (e) opp-covtype-bce (f) opp-HIGGS-bce

(g) amp-ijcnn1-rr (h) amp-covtype-rr (i) amp-HIGGS-rr (j) amp-ijcnn1-bce (k) amp-covtype-bce (l) amp-HIGGS-bce

(m) fixed-ijcnn1-rr (n) fixed-covtype-rr (o) fixed-HIGGS-rr (p) fixed-ijcnn1-bce (q) fixed-covtype-bce (r) fixed-HIGGS-bce

Figure 1: Behavior of optimization under an ϵ-AGP oracle

k with probability at least $1-\delta$. We thus determine the value of m to be:

$$m = O\left(\left(\frac{B_0 + LR}{\tau/R}\right)^2 \log\frac{K}{\delta}\right) = \tilde{O}\left(\frac{(B_0+LR)^2R^2}{\tau^2}\right)$$

where $\tilde{O}(.)$ hides factors logarithmic to $L, R, 1/\tau$, and $1/\delta$.

We can now conclude that in total $m \cdot K$ queries suffice for the server to guarantee a sub-optimality gap at most τ with probability at least $1-\delta$. Plugging in the values of K and m yields the bound claimed in (4).

5 Experiments

We used three real-world datasets from the LIBSVM library[3]: ijcnn1, covtype, and HIGGS; see the full version [Sangsiri and Tao, 2026]for details on the normalization and cleansing steps taken. Each dataset consists of n pairs (x_i, y_i) ($i \in [n]$), where $x_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \{0,1\}$ is a binary label. The values of (n,d) for ijcnn1, covtype, and HIGGS are (141691, 22 + 1), (581012, 54 + 1), and (11m, 24 + 1), respectively; see the full version [Sangsiri and Tao, 2026] for the meaning of "+1". Each pair defines a local loss function $\ell_i(w)$ where $w \in \mathbb{R}^d$. The goal of optimization is to minimize the function $f(w)$ defined in (1). We employed two loss functions for each $i \in [n]$: (i) *robust regression* (RR), $\ell_i(w) = (\sigma(\langle w, x_i\rangle) - y_i)^2$, where $\sigma : \mathbb{R} \to (0,1)$ is defined as $\sigma(z) = 1/(1+e^{-z})$; and (ii) *binary cross-entropy* (BCE), $\ell_i(w) = -y_i \log(\sigma(\langle w, x_i\rangle)) - (1-y_i)\log(1-\sigma(\langle w, x_i\rangle))$. Each ℓ_i is a convex and L_i-smooth function, where the value of L_i was calculated based on (x_i, y_i). The value L (in our problem definition) was determined as $\max_{i=1}^n L_i$.

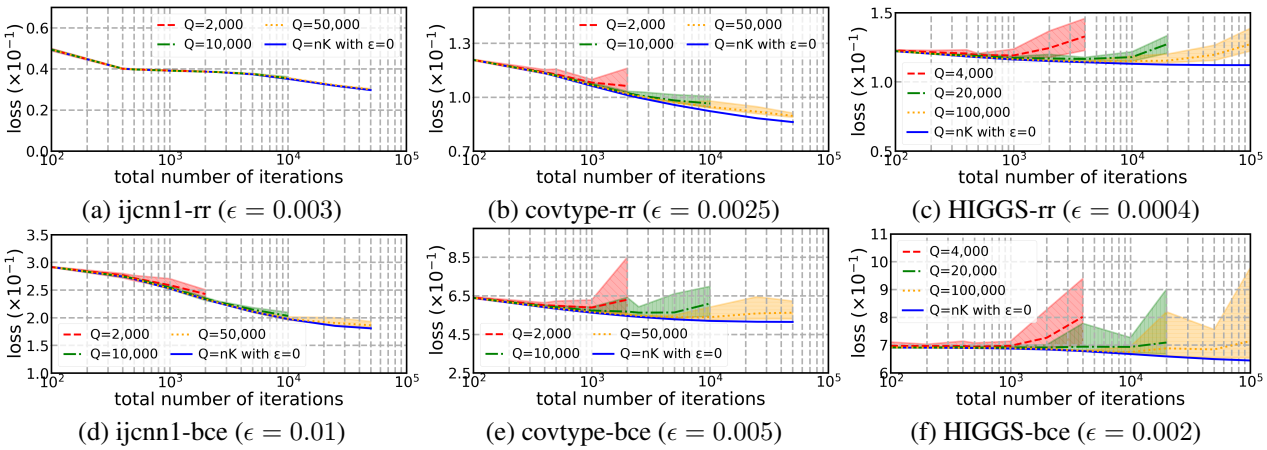
[3]Https://www.csie.ntu.edu.tw/~cjlin/libsvm

Optimization under AGP Oracles. The first set of experiments studies the convergence behavior of the `AGP-opt` algorithm in Section 3.3. We examined three ϵ-AGP oracles:

- *Opposing:* the oracle perturbs $\nabla f(w)$ by distance ϵ toward the direction *opposite* to that of $\nabla f(w)$;
- *Amplifying:* the oracle perturbs $\nabla f(w)$ by distance ϵ in the same direction as $\nabla f(w)$;
- *Fixed-Direction:* the oracle perturbs $\nabla f(w)$ by distance ϵ toward a fixed direction, chosen to be the negative direction of the first dimension of $\mathbb{R}^d$.

Figure 1 plots the $f(w_k)$ value (labeled as "loss" in the diagrams) as a function of k (the current iteration number) for four ϵ values of different magnitudes: $\epsilon = 0$ represents conventional gradient descent, and the other three values form a geometric progression. The caption of each diagram indicates the oracle type, dataset, and local loss functions; e.g., opp-ijcnn1-rr means that oracle is opposing, the dataset is ijcnn1, and all $\ell_1, \ell_2, ..., \ell_n$ adopt the RR function. We disabled the "early termination" condition at Line 4 of `AGP-opt`. In theory, the condition is needed because when $\|\nabla f(w)\|$ drops to near ϵ, the perturbation of an ϵ-AGP oracle becomes significant and, in the worst case, can be made to harm optimization (as discussed in the full version [Sangsiri and Tao, 2026], the provable sub-optimality gap is higher without such a condition). In practice, however, such worst-case scenarios rarely happen, thus allowing further optimization even beyond the moment of early termination.

The behavior of an opposing oracle in Figures 1(a)-(f) contrasts that of an amplifying oracle in Figures 1(g)-(l). In essence, an opposing oracle suppresses the "step length" of optimization, while an amplifying oracle strengthens it. Un-



(a) ijcnn1-rr ($\epsilon = 0.003$) (b) covtype-rr ($\epsilon = 0.0025$) (c) HIGGS-rr ($\epsilon = 0.0004$)

(d) ijcnn1-bce ($\epsilon = 0.01$) (e) covtype-bce ($\epsilon = 0.005$) (f) HIGGS-bce ($\epsilon = 0.002$)

Figure 2: Effects of query allocation in distributed learning

der an opposing oracle, higher ϵ values incurred larger suboptimality gaps, but the opposite was observed under an amplifying oracle. This seemingly surprising phenomenon is in fact consistent with the understanding that a step length chosen by theoretical analysis — $\frac{1}{2L}\texttt{ora}(w_k)$ for `AGP-opt` — is often conservatively small, such that an amplifying oracle "happened to" perform optimization at a better rate.

Figures 1(m)-(r) show that, under a fixed-direction oracle, the red curve — representing a relatively large ϵ value — exhibited a V-shape. To explain why, first note that at the early stages of optimization, the real gradient $\nabla f(w_k)$ has a large norm such that perturbation has little effect. As the algorithm approaches the optimal vector w^*, the magnitude of $\|\nabla f(w_k)\|$ diminishes, i.e., only small updates are intended. In this case, perturbation dictates the update direction, pulling the algorithm past w^* and thus causing the loss to go up. Eventually, the optimization converges toward the w where $\nabla f(w) = (\epsilon, 0, 0, ..., 0)^{\mathsf{T}}$. In fact, for small ϵ values such V-shapes also exist but they are difficult to observe.

Query Budget Allocation in Distributed Learning. The second set of experiments investigates distributed learning under a specific query budget Q, i.e., the number of queries that the server can afford to issue. We consider that the server allocates the budget as follows. First, it chooses a parameter $K \leq Q$, which is the *total* number of iterations of `AGP-opt` (without early termination) to perform. Then, in each iteration, it issues $m = Q/K$ queries to compute $\texttt{ora}(w_k)$ as in (22). There is a tradeoff: K needs to be sufficiently large for `AGP-opt` to do enough iterations to approach w^*, but increasing K reduces m, which elevates the uncertainty in center estimation (see Lemma 5). Our purpose is to provide a guideline for choosing K properly.

Figure 2 plots the final outcome $f(w_K)$, the value of f after K iterations of `AGP-opt`, as a function of K; we will refer to $f(w_K)$ as the *final loss*. The caption under a diagram indicates the dataset, local loss functions, and ϵ value deployed. In these experiments, when queried with w, a client $i \in [n]$ perturbs $\nabla \ell_i(w)$ using one of the three strategies described earlier — opposing, amplifying, or fixed-direction perturbation — with equal probability. Each diagram includes 4 curves: the orange, green, and red curves are obtained with geometrically-increasing values of Q, as well as a blue curve, marked as "$Q = nK$ with $\epsilon = 0$", corresponding to the "unrealistic" method where the server simply queries every client in each iteration and each client performs no perturbation at all. The blue curve serves as a reference for the best achievable final loss at each K. To account for the randomness in the strategy described in Section 4.2, we use shaded regions to report the range of final loss values across multiple runs.

Evidently, K must not be too small: optimization through gradient-based adjustments *does* require many steps. In Figures 2(a) and 2(d), the best K turned out to be Q, i.e., only one query was issued per iteration. However, this is not representative: in Figures 2(b), 2(c), 2(e), and 2(f), the shaded regions start to grow considerably after Q/K has dropped below a certain threshold. On the other hand, when $Q/K \geq 100$, shaded regions are hardly noticeable in all experiments. Furthermore, at the same K, as long as $Q/K \geq 100$, increasing Q provided little help in lowering the final loss value. The above observations suggest that $m = 100$ queries per iteration are adequate for reliable center estimation, because of which $K = Q/100$ is a robust choice in reality.

6 Conclusions

We have characterized distributed optimization under adversarial gradient perturbations, a setting in which each client's gradient reply to a server query may arbitrarily deviate from the true gradient, subject only to a known distortion bound. Our work addresses two fundamental questions: (i) what level of optimization accuracy can be guaranteed in the presence of such adversarial replies, and (ii) how many gradient queries are sufficient to reach a given accuracy level. We establish sharp feasibility thresholds on achievable accuracy, showing a strict regime in which no algorithm can guarantee meaningful progress, as well as a complementary regime in which reliable optimization is always possible. For the attainable regime, we develop algorithms that provably reach the target accuracy using a bounded number of queries. Future directions include proving tight bounds on the query complexity and extending this framework to richer function classes.

References

[Agarwal *et al.*, 2012] Alekh Agarwal, Peter L. Bartlett, Pradeep Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

[Alistarh *et al.*, 2018] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 4618–4628, 2018.

[Blanchard *et al.*, 2017] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 119–129, 2017.

[Boob *et al.*, 2023] Digvijay Boob, Qi Deng, and Guanghui Lan. Stochastic first-order methods for convex and nonconvex functional constrained optimization. *Mathematical Programming*, 197(1):215–279, 2023.

[Bubeck, 2015] Sebastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[Chai *et al.*, 2023] Chengliang Chai, Jiayi Wang, Nan Tang, Ye Yuan, Jiabin Liu, Yuhao Deng, and Guoren Wang. Efficient coreset selection with cluster-based methods. In *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*, pages 167–178, 2023.

[Chang *et al.*, 2022] Fu-Chieh Chang, Farhang Nabiei, Pei-Yuan Wu, Alexandru Cioba, Sattar Vakili, and Alberto Bernacchia. Gradient descent: Robustness to adversarial corruption. In *Proceedings of International OPT Workshop on Optimization for Machine Learning*, 2022.

[d'Aspremont, 2008] Alexandre d'Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008.

[Dekel *et al.*, 2012] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research (JMLR)*, 13:165–202, 2012.

[Devolder *et al.*, 2014] Olivier Devolder, Francois Glineur, and Yurii E. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[Garrigos and Gower, 2024] Guillaume Garrigos and Robert M. Gower. Handbook of convergence theorems for (stochastic) gradient methods. *ArXiv*, abs/2301.11235, 2024.

[Hallak and Levy, 2024] Nadav Hallak and Kfir Yehuda Levy. A study of first-order methods with a deterministic relative-error gradient oracle. In *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[Huang *et al.*, 2021] Jiawei Huang, Ruomin Huang, Wenjie Liu, Nikolaos M. Freris, and Hu Ding. A novel sequential coreset method for gradient descent algorithms. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 4412–4422, 2021.

[Izzo *et al.*, 2022] Zachary Izzo, James Zou, and Lexing Ying. How to learn when data gradually reacts to your model. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3998–4035, 2022.

[Jaggi, 2013] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 427–435, 2013.

[Karimireddy *et al.*, 2021] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 5311–5319, 2021.

[Kerger *et al.*, 2024] Phillip A. Kerger, Marco Molinaro, Hongyi Jiang, and Amitabh Basu. A universal transfer theorem for convex optimization algorithms using inexact first-order oracles. In *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[Kohler and Lucchi, 2017] Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for nonconvex optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1895–1904, 2017.

[Lan, 2020] Guanghui Lan. *First-order and Stochastic Optimization Methods for Machine Learning*. Springer, 2020.

[Lei and Tang, 2021] Yunwen Lei and Ke Tang. Learning rates for stochastic gradient descent with nonconvex objectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 43(12):4505–4511, 2021.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[Mirzasoleiman *et al.*, 2020] Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 6950–6960, 2020.

[Nemirovski *et al.*, 2009] Arkadi Nemirovski, Anatoli B. Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[Nemirovsky and Yudin, 1983] Arkadi Nemirovsky and David Berkovich Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, 1983.

[Nesterov, 2012] Yurii E. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[Sangsiri and Tao, 2026] Nawapon Sangsiri and Yufei Tao. Distributed learning with adversarial gradient perturbations. *CoRR*, abs/2605.03313, 2026.

[Tsybakov, 2003] Alexandre B. Tsybakov. Optimal rates of aggregation. In Bernhard Scholkopf and Manfred K. Warmuth, editors, *Proceedings of Conference on Learning Theory (COLT)*, volume 2777, pages 303–313, 2003.

[Wang and Tan, 2024] Shuche Wang and Vincent Y. F. Tan. Robust distributed gradient descent to corruption over noisy channels. In *IEEE International Symposium on Information Theory*, pages 2520–2525, 2024.

[Wang and Tan, 2025] Shuche Wang and Vincent Y. F. Tan. A mirror descent-based algorithm for corruption-tolerant distributed gradient descent. *IEEE Transactions on Signal Processing*, 73:827–842, 2025.

[Yin *et al.*, 2018] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 5636–5645, 2018.