Gradient Descent: Robustness to Adversarial Corruption

Fu-Chieh Chang[1,2] MARK-FC.CHANG@MTKRESEARCH.COM
Farhang Nabiei[2] FARHANG.NABIEI@MTKRESEARCH.COM
Pei-Yuan Wu[1] PEIYUANWU@NTU.EDU.TW
Alexandru Cioba[2] ALEXANDRU.CIOBA@MTKRESEARCH.COM
Sattar Vakili[2] SV388@CORNELL.EDU
Alberto Bernacchia[2] ALBERTO.BERNACCHIA@MTKRESEARCH.COM
[1]*Graduate Institute of Communication Engineering*
National Taiwan University
Taipei, Taiwan
[2]*MediaTek Research*
Cambridge, UK

Abstract

Optimization using gradient descent (GD) is a ubiquitous practice in various machine learning problems including training large neural networks. Noise-free GD and stochastic GD–corrupted by random noise–have been extensively studied in the literature, but less attention has been paid to an adversarial setting, that is subject to adversarial corruptions in the gradient values. In this work, we analyze the performance of GD under a proposed general adversarial framework. For the class of functions satisfying the Polyak-Łojasiewicz condition, we derive finite time bounds on a minimax optimization error. Based on this bound, we provide a guideline on the choice of learning rate sequence with theoretical guarantees on the robustness of GD against adversarial corruption.

1. Introduction

Optimization using gradient descent (GD) is central to training machine learning models. In this work, we consider the dynamics of GD under adversarial corruptions, which to our knowledge is a novel formulation. Optimization using corrupted gradients has been studied in many forms [e.g., 2, 7, 15], as discussed in section 1.2 depending on the practical problem of interest. Contrary to these studies, our framework is not limited to a specific real-world application, but rather attempts to understand the robustness of gradient descent under the worst possible corruption. In this framework, at each GD iteration n, an adversary corrupts the values of the gradients, up to a budget constraint. We refer to this framework as adversarial GD (Adv-GD). The Adv-GD framework yields a mini-max performance measure: the optimization error minimized by GD and maximized by the adversary; that can also be interpreted as a game between the optimization algorithm and the adversary. Compared to stochastic GD with random corruptions, adversarial corruptions are more challenging to analyze for at least two reasons. In stochastic GD, it is typically assumed that the noisy gradient is an unbiased estimator of the true gradient. Adversarial corruptions, however, may introduce bias to the GD dynamics which are more challenging to keep track of. In addition, in stochastic GD, typically a constant upper bound on the variance of the noisy gradients is as-

© F.-C. Chang[1,2], F. Nabiei[2], P.-Y. Wu[1], A. Cioba[2], S. Vakili[2] & A. Bernacchia[2].

sumed. In Adv-GD framework, however, the adversary is allowed to spend an arbitrary amount of its budget, which can be relatively large, to corrupt a gradient observation.

In this work, we focus on the objective functions satisfying the PL condition, which is a very general and practically relevant class of functions, including many machine learning models, e.g., overparameterized neural networks [e.g., see, 5]. Our contributions can be summarized as follows:

- We introduce a novel framework for the analysis of GD under adversarial corruptions, and prove finite time minimax error bounds for the class of functions satisfying the PL condition. (see, Section 2)
- We generalize the measure of corruption from l^2 norm —the case of variance— to arbitrary l^q norms, with $q \geq 1$, which may be of broader interest. (see, Section 2)
- We provide a guideline on the choice of learning rate sequence with theoretical guarantees on the robustness of GD algorithm against adversarial corruption. (see, Section 3)

1.1. Motivation

We provide a general framework for evaluating the robustness of GD to adversarial corruption. In typical analytical results for GD, the corruption is modelled as a stochastic noise (usually with bounded variance), and the convergence is given in expectation. Alternatively, we provide a worst-case error under a corruption budget assumption. This interpretation of convergence can be generally applied to all the use cases of GD as an upper bound on the optimization error when the nature of corruption is unknown.

In many modern machine learning applications, such as in IoT and federated learning, the gradients of the model are communicated to a server [3, 11]. The gradients may be arbitrarily corrupted due to communication issues. The communication may also be intercepted maliciously with the aim of adversarially corrupting the performance. In these cases the assumption of a stochastic corruption is not suitable. Our results provide bounds on the convergence error of GD based on the amount of corruptions. In simple words, we characterize the amount of corruption in gradient values (in terms of l^q norm) that is tolerable for GD optimization.

1.2. Related Work

A different approach to adversarial optimization has been considered under the well-established online learning literature. In this approach, a sequence of typically convex objective functions are selected by an adversary. The performance is measured as the total loss compared to the cumulative value of the objective functions evaluated at a single point. Several gradient based algorithms [e.g., *online GD*, 8, 9, 19], and non gradient based algorithms [e.g., *follow the (perturbed) leader*, 1, 10] are introduced in this setting. See [13, 16] for surveys. In contrast, in Adv-GD framework, the adversary directly corrupts the gradient values. In addition, the adversary can allocate the corruptions arbitrarily (see the minimax optimization error defined in (3)). With any growing budget, this violates the assumptions in the online learning setting. Due to their inherently different assumptions, our results cannot be directly compared with the ones given in the online learning setting.

Several existing work considered GD under adversarial corruption of gradients. The work of [2] considered *Byzantine* stochastic GD in a distributed stochastic optimization setting, where a fraction of machines which compute stochastic gradients in every iteration are Byzantine, and may behave

adversarially. In this setting, the main focus of the problem is on detecting Byzantine machines in the presence of available true gradient values. Similarly, [7] and [15] considered an adversary corrupting a fraction of sample gradients. Their work focused on detecting the corrupted samples as outliers. This line of work is different from ours in that, in our setting, the adversary is allowed to arbitrarily allocate the corruptions through the optimization sequence, leading to a minimax (worst case) performance measure.

2. Gradient Descent with Adversarial Corruptions

In this section, we present the Adv-GD framework, as well as other relevant notations and definitions. Consider the problem of minimizing a differentiable objective function $f : \mathcal{X} \to \mathbb{R}$, where the domain $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$. The goal is to get as close as possible to a global minimum $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. Recall the standard iterative updating rule of (stochastic) GD:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \eta_k \tilde{\mathbf{g}}(\mathbf{x}_{k-1}), \tag{1}$$

where $k = 1, 2, \ldots$ is a discrete iteration index, the iterations start at an arbitrary (possibly random) initial point $\mathbf{x}_0$, and $(\eta_k \in \mathbb{R})_{k=1}^{\infty}$ is the sequence of learning rates. In vanilla GD, we have simply $\tilde{\mathbf{g}}(\mathbf{x}_{k-1}) = \mathbf{g}(\mathbf{x}_{k-1})$, using the notation $\mathbf{g}(\cdot) = \nabla f(\cdot)$ for the gradient. In stochastic GD, $\tilde{\mathbf{g}}(\mathbf{x}_{k-1})$ is typically assumed to be an unbiased estimator of $\mathbf{g}(\mathbf{x}_{k-1})$ with bounded variance [e.g., see, 6, and references therein]. That is, in stochastic GD, we have randomly corrupted gradient estimates $\tilde{\mathbf{g}}(\mathbf{x}_{k-1}) = \mathbf{g}(\mathbf{x}_{k-1}) + \boldsymbol{\xi}_k$, where $\boldsymbol{\xi}_k$ is a zero mean random observation noise at iteration k, with bounded variance, independent and identically distributed over iterations.

In the Adv-GD framework, we consider adversarially corrupted gradient values. Specifically, at each iteration k, the optimizer asks for the value of gradient $\mathbf{g}(\mathbf{x}_{k-1})$. The gradient values, however, may be adversarially corrupted, so that the optimizer receives $\tilde{\mathbf{g}}(\mathbf{x}_{k-1}) \in \mathbb{R}^d$. We are interested in the dynamics of GD update rule (1), under this setting.

Budget Constraint: With no constraints, the setting is not interesting, as the adversarial corruptions can set the sequence $(\mathbf{x}_k)_{k=1}^{\infty}$ to any values in $\mathbb{R}^d$. It is therefore natural to set a budget constraint on adversarial corruptions. In particular, consider a distance measure $\mathcal{D}(.,.) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$. Let $c(k) := \mathcal{D}(\mathbf{g}(\mathbf{x}_{k-1}), \tilde{\mathbf{g}}(\mathbf{x}_{k-1}))$ quantify the amount of corruption in the gradient value, allocated by the adversary, at iteration k. Then, the total corruption budget $b : \mathbb{N} \to \mathbb{R}_{>0}$ imposed on the adversary is a positive and non-decreasing sequence limiting the amount of corruptions:

$$\left\| [c(k)]_{k=1}^{n} \right\|_{l^p} \leq b(n), \quad \forall n \in \mathbb{N}, \tag{2}$$

where $\|\cdot\|_{l^p}$ denotes the l^p norm. We use the notation $\|\cdot\|$ exceptionally in the case of l^2 norm. Given a budget, let us define the set of *admissible* corruptions as follows.

Definition 1 *For a given budget b and time n, the set of admissible corruptions $\mathcal{C}_{b,p}^n$ is defined as*

$$\mathcal{C}_{b,p}^n = \left\{ (c_k \in \mathbb{R}_{\geq 0})_{k=1}^{n} : \forall\, n' \leq n, \left\| [c_k]_{k=1}^{n'} \right\|_{l^p} \leq b(n') \right\}.$$

Performance Measure: In the Adv-GD framework, the finite time performance of the optimizer is measured in terms of a minimax optimization error; minimized by the optimizer and maximized by the adversary. Specifically, at each iteration n, the minimax optimization error is defined as

$$r_n = \sup_{(\boldsymbol{\xi}_k)_{k=1}^n : (c(k))_{k=1}^n \in \mathcal{C}_{b,p}^n} (f(\mathbf{x}_n) - f(\mathbf{x}^*)), \tag{3}$$

where $\xi_k = \tilde{\mathbf{g}}(\mathbf{x}_{k-1}) - \mathbf{g}(\mathbf{x}_{k-1})$. Recall that $\mathbf{x}_n$ is the point after n-th iteration in the GD update rule, $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ is a global minimum, and $c(k) = \mathcal{D}\left(\mathbf{g}(\mathbf{x}_{k-1}), \tilde{\mathbf{g}}(\mathbf{x}_{k-1})\right)$ is the amount of corruption at iteration k. The $\sup$ is taken over all admissible corruption sequences.

Game Theoretic Interpretation: The Adv-GD framework can be interpreted as a zero-sum game between the optimizer and the adversary. Before starting the GD iterations, the optimizer selects the learning rate sequence $(\eta_k)_{k=1}^n$. With the knowledge of the learning rate sequence, the adversary selects the gradient corruption sequence $(\boldsymbol{\xi}_k)_{k=1}^n$ subject to the admissibility constraint: $(c(k))_{k=1}^n \in \mathcal{C}_{b,p}^n$. At the end of n iterations, the optimizer incurs a loss of $f(\mathbf{x}_n) - f(\mathbf{x}^*)$, while the adversary incurs a loss of $f(\mathbf{x}^*) - f(\mathbf{x}_n)$. The goal of each player is to minimize their own loss.

Remark 2 *In a zero-sum game where the adversary has no knowledge of the learning rate schedule, it would choose a gradient corruption sequence which may cause lower optimization error compared to the adversary with the knowledge. Hence, the minimax optimization error measured in our Adv-GD framework can naturally be an upper bound of the optimization error in this scenario.*

Corruption Measure: We measure the corruption at each iteration, using l^q norm of the distance between the true and corrupted gradient values. In particular, in our analysis, we consider

$$\mathcal{D}(\mathbf{g}(\mathbf{x}_{k-1}), \tilde{\mathbf{g}}(\mathbf{x}_{k-1})) = \|\boldsymbol{\xi}_k\|_{l^q}.$$

In the case of stochastic GD, a typical assumption is the boundedness of the variance of the noisy gradients at each iteration; $\mathbb{E}[\|\boldsymbol{\xi}_k\|^2]$ is assumed bounded. This can be interpreted as measuring the random corruptions in terms of l^2 norm. In this sense, we generalize the typical measure of corruption from l^2 norm to l^q norm (in addition to considering adversarial corruptions, in contrast to random ones). Considering also the freedom in choosing p for the norm of corruption sequence in eq. 2, we investigate a quite general setting of gradient corruption in gradient descent. We present our results for the class of functions satisfying the PL condition, with additional assumptions:

Definition 3 (PL Condition) *We say that a function satisfies the PL condition with parameter $\alpha \in \mathbb{R}_{>0}$, if $\|\mathbf{g}(\mathbf{x})\|^2 \geq \alpha(f(\mathbf{x}) - f(\mathbf{x}^*)), \quad \forall \mathbf{x} \in \mathcal{X}$. Recall the notation $\mathbf{g}(\cdot) = \nabla f(\cdot)$. We use the notation $\mathcal{F}_{PL}^{\alpha}$ to denote the class of functions satisfying the PL condition with parameter α.*

Definition 4 (Strong Convexity) *We say that a function is strongly convex with parameter μ, if $f(\mathbf{x}') \geq f(\mathbf{x}) + \langle \mathbf{g}(\mathbf{x}), \mathbf{x}' - \mathbf{x}\rangle + \frac{\mu}{2}\|\mathbf{x}' - \mathbf{x}\|^2, \quad \forall \mathbf{x} \in \mathcal{X}$.*

Definition 5 (Smoothness) *We say that a function is smooth with parameter β, if $f(\mathbf{x}') \leq f(\mathbf{x}) + \langle \mathbf{g}(\mathbf{x}), \mathbf{x}' - \mathbf{x}\rangle + \frac{\beta}{2}\|\mathbf{x}' - \mathbf{x}\|^2, \quad \forall \mathbf{x} \in \mathcal{X}$. We use the notation $\mathcal{F}_S^{\beta}$ to denote the class of smooth functions with parameter β.*

$\mathcal{X}$	Domain	r_n	Minimax optimization error	$a \vee b$	$\max\{a, b\}$
f	Objective function	$b(n)$	Budget up to time n included	$a \wedge b$	$\min\{a, b\}$
$(\mathbf{x}_k)_{k=1}^n$	Iteration indexed points in domain	$c(k)$	Budget spent at time k	a^+	$\max(a, 0)$
α	PL parameter	$\tilde{\mathbf{g}}(\cdot)$	Corrupted gradient	$\mathbf{g}(\cdot)$	Gradient
μ	Strong convexity parameter	$\boldsymbol{\xi}_k$	Corruption at time k	η_k	Learning rate
β	Smoothness parameter	γ	Upper bound on gradient norm		

Table 1: Notations

3. Minimax Error Bounds under the Adv-GD Framework

In this section, we present our main result on the finite time minimax (worst-case) optimization error bound under the Adv-GD framework. We prove error bounds which depend on the error at the initialization and the corruption budget.

Theorem 6 (Adv-GD: Finite Time Optimization Error) Consider the Adv-GD framework formalized in Section 2. Assume $f \in \mathcal{F}_{\text{PL}}^{\alpha} \cap \mathcal{F}_{\text{S}}^{\beta}$, and $\|\mathbf{g}(\mathbf{x})\| \leq \gamma$ for all $\mathbf{x} \in \mathcal{X}$. Let $\eta_k = \left(\frac{2}{\alpha}\left(1 - (\frac{k}{k+1})^{\omega}\right) \wedge \frac{1}{\beta}\right)$ for a fixed $\omega \geq 1$. Define $n_0 = \min\left\{n \in \mathbb{N} : \frac{2}{\alpha}\left(1 - \left(\frac{n}{n+1}\right)^{\omega}\right) \leq \frac{1}{\beta}\right\}$. Define $d_q = d^{(\frac{1}{2}-\frac{1}{q})^+}$. We have the following bound on the minimax optimization error defined in (3).

$$\boxed{r_n \leq A_n \left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right) + B_{p,n} b(n) + C_{p,n}(b(n))^2,}$$

where $(A_n \in \mathbb{R}_{>0})_{n=1}^{\infty}$, $(B_{p,n} \in \mathbb{R}_{>0})_{n=1}^{\infty}$ and $(C_{p,n} \in \mathbb{R}_{>0})_{n=1}^{\infty}$ are sequences satisfying:

- ◊ For $n < n_0$: $A_n = (1 - \frac{\alpha}{2\beta})^n$, $B_{p,n} = 0$, $C_{p,n} \leq \frac{d_q^2}{2\beta}$ when $1 \leq p \leq 2$, and $C_{p,n} \leq \frac{d_q^2}{\alpha}$ when $p > 2$.
- ◊ For $n \geq n_0$: $A_n \in \mathcal{O}(n^{-\omega})$, $B_{p,n} \in \mathcal{O}(d_q n^{-\frac{1}{p}})$, and

$$\begin{cases} C_{p,n} \in \mathcal{O}\left(d_q^2 n^{-(\omega \wedge 2)}\right), & \text{when } 1 \leq p \leq 2 \text{ and } \omega \geq 1, \\ C_{p,n} \in \mathcal{O}\left(d_q^2 n^{-\omega}\right), & \text{when } p > 2 \text{ and } 1 \leq \omega < 1 + 2/p, \\ C_{p,n} \in \mathcal{O}\left(d_q^2 n^{-\omega} (\log n)^{1-2/p}\right), & \text{when } p > 2 \text{ and } \omega = 1 + 2/p, \\ C_{p,n} \in \mathcal{O}\left(d_q^2 n^{-1-2/p}\right), & \text{when } p > 2 \text{ and } \omega > 1 + 2/p. \end{cases}$$

Proofs and further details are provided in the Appendix A. Theorem 6 provides very general conditions for the robustness of GD under adversarial corruptions, covering various measures for the corruptions in terms of l^p norm of cumulative effect of corruptions over iterations, and l^q norm of corruption at each iteration. In Corollary 7, we specialize this theorem for the case of $p = 1$, that corresponds to simply adding the corruptions over iterations $\left\|[c(k)]_{k=1}^N\right\|_{l^1} = \sum_{k=1}^n c(k)$.

Corollary 7 (Adv-GD: Finite Time Optimization Error with $p = 1$) Under the setting of Theorem 6, set $\omega = 2$ which amounts to $\eta_n = \left(\frac{2}{\alpha}\left(1 - (\frac{n}{n+1})^2\right) \wedge \frac{1}{\beta}\right) \sim \frac{1}{n}$. We then have the following. For the case of $p = 1$:

$$f(x_n) - f(x^*) \quad \in \mathcal{O}\left(\frac{f(x_0) - f(x^*)}{n^2} + \frac{d_q b(n)}{n} + \frac{(d_q b(n))^2}{n^2}\right).$$

Recall $\mathcal{D}\left(\mathbf{g}(\mathbf{x}_{k-1}), \tilde{\mathbf{g}}(\mathbf{x}_{k-1})\right) = \|\boldsymbol{\xi}_k\|_{l^q}$. The case of $q = 2$ is similar to the typical stochastic GD setting, where, the random corruptions are subject to a bound on their expected l^2 norm. In Adv-GD framework, $d_2 = 1$, by definition. As q grows larger, d_q grows closer to $\sqrt{d}$.

Remark 8 *The particular choice of the learning rate comes from the tradeoff between corruption free optimization and removing the corruption. A larger learning rate (up to a constant value smaller than 1) is preferred for corruption free GD. That however cannot efficiently control the corruption. A very small learning rate can efficiently remove the effect of corruption. It however results in a very slow convergence for the corruption free part of the optimization. Efficiently trading off these two directions leads to our choice of learning rate. The details of this trade off can be found in our analysis in the Appendix A.*

Remark 9 *Theorem 6 readily holds true under strong convexity (replacing the PL condition). This follows from the fact that strong convexity with parameter μ implies* PL *condition with parameter $\alpha = 2\mu$.*

Numerical Evaluations We provide numerical experiments for the Adv-GD framework in Appendix B. The experimental results corroborate our theoretical findings.

4. Conclusion

We introduced the novel framework of Adv-GD, in which gradients are corrupted by an adversary, limited by a given budget. Thus, our results are applicable for the worst-case scenario of gradient corruptions that might happen in practice. We provided finite time upper bounds on the worst-case optimization error, for the family of smooth PL functions. We also show that the robustness of GD algorithm can be achieved via a proper choice of learning rate sequence based on Theorem 6 or Collorary 7. It remains unknown whether the error bounds are optimal. Future works may also look into obtaining lower bounds for the error, to compare with our upper bounds, and investigate the question of what is the optimal adversarial corruption. We believe that the novel framework proposed in this study will inspire further theoretical work to address all the open questions.

References

[1] Naman Agarwal, Alon Gonen, and Elad Hazan. Learning in non-convex games with an optimization oracle. In *Conference on Learning Theory*, pages 18–29. PMLR, 2019.

[2] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. *arXiv preprint arXiv:1803.08917*, 2018.

[3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

[4] Mihai Anitescu. Degenerate nonlinear programming with a quadratic growth condition. *SIAM Journal on Optimization*, 10(4):1116–1135, 2000.

[5] Raef Bassily, Mikhail Belkin, and Siyuan Ma. On exponential convergence of sgd in non-convex over-parametrized learning. *arXiv preprint arXiv:1811.02564*, 2018.

[6] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3–4):231–357, nov 2015. ISSN 1935-8237. doi: 10.1561/2200000050.

[7] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606. PMLR, 2019.

[8] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *The Journal of Machine Learning Research*, 15(1): 2489–2512, 2014.

[9] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

[10] Ruitong Huang, Tor Lattimore, András György, and Csaba Szepesvári. Following the leader and fast rates in linear prediction: Curved constraint sets and other regularities. *Advances in Neural Information Processing Systems*, 29, 2016.

[11] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[12] Ji Liu, Steve Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In *International Conference on Machine Learning*, pages 469–477. PMLR, 2014.

[13] H Brendan McMahan. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017.

[14] Ion Necoara, Yu Nesterov, and Francois Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175(1):69–107, 2019.

[15] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

[16] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[17] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[18] Hui Zhang and Wotao Yin. Gradient methods for convex minimization: better rates under weaker conditions. *arXiv preprint arXiv:1303.4645*, 2013.

[19] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

Appendix A. Proofs and Details of Theoretical Analysis

In this section, we provide a proof for Theorem 6. As a consequence of smoothness assumption ($f \in \mathcal{F}_{\text{S}}^{\beta}$), we have

$$\begin{aligned} f(\mathbf{x}_k) &\leq f(\mathbf{x}_{k-1}) + \langle \mathbf{g}(\mathbf{x}_{k-1}), (\mathbf{x}_k - \mathbf{x}_{k-1}) \rangle + \frac{\beta}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2 \\ &= f(\mathbf{x}_{k-1}) - \langle \mathbf{g}(\mathbf{x}_{k-1}), \eta_k \tilde{\mathbf{g}}(\mathbf{x}_{k-1}) \rangle + \frac{\beta}{2}\|\eta_k \tilde{\mathbf{g}}(\mathbf{x}_{k-1})\|^2 \\ &= f(\mathbf{x}_{k-1}) - \eta_k \langle \mathbf{g}(\mathbf{x}_{k-1}), (\mathbf{g}(\mathbf{x}_{k-1}) + \boldsymbol{\xi}_k) \rangle + (\beta\eta_k^2/2)\|\mathbf{g}(\mathbf{x}_{k-1}) + \boldsymbol{\xi}_k\|^2 \\ &= f(\mathbf{x}_{k-1}) - \eta_k(1 - \beta\eta_k/2)\|\mathbf{g}(\mathbf{x}_{k-1})\|^2 - \eta_k(1 - \beta\eta_k)\langle \mathbf{g}(\mathbf{x}_{k-1}), \boldsymbol{\xi}_k \rangle + (\beta\eta_k^2/2)\|\boldsymbol{\xi}_k\|^2. \end{aligned}$$

The second line follows from the GD update rule (1); the third line follows from the definition of $\tilde{\mathbf{g}}$; the fourth line is a rearrangement of the terms. For simplicity of notation, let us use $y_k = f(\mathbf{x}_k) - f(\mathbf{x}^*)$ and $c_k = \|\boldsymbol{\xi}_k\|_{l^q}$. Also, recall the notation $d_q = d^{(\frac{1}{2} - \frac{1}{q})^+}$. We can rearrange the last line as follows

$$\begin{aligned} y_k &\leq y_{k-1} - \eta_k(1 - \beta\eta_k/2)\|\mathbf{g}(\mathbf{x}_{k-1})\|^2 - \eta_k(1 - \beta\eta_k)\langle \mathbf{g}(\mathbf{x}_{k-1}), \boldsymbol{\xi}_k \rangle + (\beta\eta_k^2/2)\|\boldsymbol{\xi}_k\|^2 \\ &\leq y_{k-1} - \eta_k(1 - \beta\eta_k/2)\|\mathbf{g}(\mathbf{x}_{k-1})\|^2 + \gamma d_q |\eta_k(1 - \beta\eta_k)| c_k + (\beta d_q^2 \eta_k^2/2) c_k^2 \\ &\leq y_{k-1} - \alpha\eta_k(1 - \beta\eta_k/2) y_{k-1} + \gamma d_q |\eta_k(1 - \beta\eta_k)| c_k + (\beta d_q^2 \eta_k^2/2) c_k^2. \end{aligned}$$

The second line follows from $\|\cdot\| \leq d_q \|\cdot\|_{l^q}$. Note that we have $\|\cdot\| \leq \|\cdot\|_{l^q}$ for $q \leq 2$, and $\|\cdot\| \leq d^{\frac{1}{2} - \frac{1}{q}} \|\cdot\|_{l^q}$ for $q > 2$. The third line follows from the PL condition ($f \in \mathcal{F}_{\text{PL}}^{\alpha}$). Note that $\eta_k(1 - \beta\eta_k/2) \geq 0$, by definition of η_k in the statement of the theorem. Further simplifying the notation and defining

$$\begin{aligned} \alpha_k &= 1 - \alpha\eta_k(1 - \beta\eta_k/2), \\ \gamma_k &= \gamma d_q \eta_k |1 - \beta\eta_k|, \\ \beta_k &= \beta d_q^2 \eta_k^2/2, \end{aligned}$$

we can write

$$\boxed{y_k \leq \alpha_k y_{k-1} + \gamma_k c_k + \beta_k c_k^2,} \tag{4}$$

We use this recursive relation to bound y_n at iteration n based on the following lemma.

Lemma 10 *Let $(y_i \in \mathbb{R}_{\geq 0})_{i=0}^{\infty}$ be a sequence satisfying*

$$y_i \leq \alpha_i y_{i-1} + \gamma_i c_i + \beta_i c_i^2, \quad \forall i \in \mathbb{N},$$

where $\alpha_i, \gamma_i, \beta_i, c_i \in \mathbb{R}_{\geq 0}$, for all $i \in \mathbb{N}$. Define $\mathbf{c}_n = [c_1, ..., c_n]$. Then, for $p \geq 1$,

$$\boxed{y_n \leq A_n y_0 + B_{p,n}\|\mathbf{c}_n\|_p + C_{p,n}\|\mathbf{c}_n\|_p^2,} \tag{5}$$

where

$$A_n = \lambda_{0,n}, \tag{6a}$$

$$B_{p,n} = \begin{cases} \max_{1\le k\le n} \gamma_k \lambda_{k,n} & \text{, if } p = 1 \\ \left(\sum_{k=1}^{n} (\gamma_k \lambda_{k,n})^{p_1}\right)^{1/p_1} & \text{, if } p > 1 \end{cases}, \tag{6b}$$

$$C_{p,n} = \begin{cases} \max_{1\le k\le n} \beta_k \lambda_{k,n} & \text{, if } 1 \le p \le 2 \\ \left(\sum_{k=1}^{n} (\beta_k \lambda_{k,n})^{p_2}\right)^{1/p_2} & \text{, if } p > 2 \end{cases}, \tag{6c}$$

and, $\lambda_{k,n} = \prod_{i=k+1}^{n} \alpha_i$*;* p_1 *and* p_2 *are chosen in a way that* $1/p + 1/p_1 = 1$*, and* $2/p + 1/p_2 = 1$*.*

The proof of Lemma 10 is given in Section A.1. The bounds given in Theorem 6 then are derived by characterizing the coefficients $A_n, B_{p,n}, C_{p,n}$ for the choice of learning rate sequence η_k in the statement of the theorem. Further details are provided in Section A.2.

Optimality of the Bound on y_n in Lemma 10: We further prove that $A_n, B_{p,n}$ and $C_{p,n}$ given in Lemma 10 are optimal in the following sense. For all $(\tilde{A}, \tilde{B}, \tilde{C}) \in \mathbb{R}^3$ such that $\tilde{A} < A_n$, $\tilde{B} < B_{p,n}$, or $\tilde{C} < C_{p,n}$, the bound on y_n given in Lemma 10 does not hold true, with $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ replacing the corresponding coefficients. In other words, the parameters given in (6) in Lemma 10 yield the tightest coefficients to which (5) holds true for all systems of inequalities of the form of (4). This observation is formalized in the following proposition.

Proposition 11 *Fix* $p \ge 1$ *and* $n \in \mathbb{N}$*, and let* p_1*,* p_2*,* A_n*,* $B_{p,n}$*,* $C_{p,n}$*,* $(\alpha_i)_{i=1}^{\infty}$*,* $(\gamma_i)_{i=1}^{\infty}$*,* $(\beta_i)_{i=1}^{\infty}$ *be defined as in Lemma 10. For all* $(\tilde{A}, \tilde{B}, \tilde{C}) \in \mathbb{R}^3$ *such that* $\tilde{A} < A_n$*,* $\tilde{B} < B_{p,n}$*, or* $\tilde{C} < C_{p,n}$*, there exist sequences* $(y_i \in \mathbb{R}_{>0})_{i=0}^{\infty}$ *and* $(c_i \in \mathbb{R}_{>0})_{i=1}^{\infty}$ *satisfying the recursive relation* $y_i \le \alpha_i y_{i-1} + \gamma_i c_i + \beta_i c_i^2$*,* $\forall i \in \mathbb{N}$*, and violating* (5)*, with* $\tilde{A}, \tilde{B}$*, and* $\tilde{C}$ *replacing the corresponding coefficients:*

$$y_n > \tilde{A} y_0 + \tilde{B} \|\mathbf{c}_n\|_p + \tilde{C} \|\mathbf{c}_n\|_p^2, \tag{7}$$

where $\mathbf{c}_n = (c_1, ..., c_n)$ *similar to Lemma 10.*

Proof is provided in Section A.3. We emphasize that Proposition 11 considers the cases where only one of the coefficients $\tilde{A}$, $\tilde{B}$ or $\tilde{C}$ is smaller than the corresponding one in Lemma 10, and not all of them together; the latter would be a weaker statement.

In Lemma 10, we established a recursive relation on the optimization error $y_k = f(\mathbf{x}_k) - f(\mathbf{x}^*)$, as follows

$$\boxed{y_k \le \alpha_k y_{k-1} + \gamma_k c_k + \beta_k c_k^2,}$$

using the following notations

$$\alpha_k = 1 - \alpha\eta_k(1 - \beta\eta_k/2), \;\; \gamma_k = \gamma d_q \eta_k |1 - \beta\eta_k|, \;\; \beta_k = \beta d_q^2 \eta_k^2/2. \tag{8}$$

Furthermore, in Lemma 10, we established a bound on y_n. In this section, we provide a proof of Lemma 10, the details of analysis for the learning rate $\eta_k = \left(\frac{2}{\alpha}\left(1 - (\frac{k}{k+1})^{\omega}\right) \wedge \frac{1}{\beta}\right)$ as a consequence of Lemma 10, as well as a proof of Proposition 11.

A.1. Proof of Lemma 10 (The Bound on y_n Based on the Recursive Relation for y_k)

It is clear that

$$y_n - \lambda_{0,n} y_0 = \sum_{k=1}^{n} \lambda_{k,n}(y_k - \alpha_k y_{k-1}) \leq \sum_{k=1}^{n} \lambda_{k,n}(\gamma_k c_k + \beta_k c_k^2) = \langle \mathbf{u}_n, \mathbf{c}_n \rangle + \langle \mathbf{v}_n, \mathbf{d}_n \rangle$$

where $\mathbf{u}_n = (\lambda_{k,n}\gamma_k)_{k=1}^n$, $\mathbf{v}_n = (\lambda_{k,n}\beta_k)_{k=1}^n$, and $\mathbf{d}_n = (c_1^2, ..., c_n^2)$. It follows by Hölder's inequality that $\langle \mathbf{u}_n, \mathbf{c}_n \rangle \leq \|\mathbf{u}_n\|_{p_1} \|\mathbf{c}_n\|_p$, and

$$\langle \mathbf{v}_n, \mathbf{d}_n \rangle \leq \begin{cases} \|\mathbf{v}_n\|_\infty \|\mathbf{d}_n\|_1 = \|\mathbf{v}_n\|_\infty \|\mathbf{c}_n\|_2^2 \leq \|\mathbf{v}_n\|_\infty \|\mathbf{c}_n\|_p^2 & \text{, if } 1 \leq p \leq 2 \\ \|\mathbf{v}_n\|_{p_2} \|\mathbf{d}\|_{p/2} = \|\mathbf{v}_n\|_{p_2} \|\mathbf{c}_n\|_p^2 & \text{, if } p > 2 \end{cases}.$$

This proves the lemma.

A.2. The Analysis of Error Bound for the Choice of η_k in Theorem 6

Recall

$$\eta_k = \frac{2}{\alpha}\left(1 - \left(\frac{k}{k+1}\right)^\omega\right) \wedge \frac{1}{\beta}, \quad n_0 = \min\left\{k \in \mathbb{N} : \frac{2}{\alpha}\left(1 - \left(\frac{k}{k+1}\right)^\omega\right) \leq \frac{1}{\beta}\right\}.$$

It follows by (8) that

$$\begin{array}{llllll} \eta_k = \frac{1}{\beta}, & \alpha_k = 1 - \frac{\alpha}{2\beta}, & \gamma_k = 0, & \beta_k = \frac{1}{2\beta} d_q^2 & \text{, if } 1 \leq k < n_0 \\ \eta_k = \frac{2}{\alpha}\left(1 - \left(\frac{k}{k+1}\right)^\omega\right), & \alpha_k \leq \left(\frac{k}{k+1}\right)^\omega, & \gamma_k \leq \gamma d_q \eta_k & \beta_k = \frac{\beta}{2} d_q^2 \eta_k^2 & \text{, if } k \geq n_0 \end{array}.$$

Hence $\lambda_{k,n} = \prod_{i=k+1}^n \alpha_i$ is given by

$$\begin{array}{ll} \lambda_{k,n} = \left(1 - \frac{\alpha}{2\beta}\right)^{n-k} & \text{, if } 0 \leq k \leq n < n_0 \\ \lambda_{k,n} \leq \left(1 - \frac{\alpha}{2\beta}\right)^{n_0-k-1} \left(\frac{n_0}{n+1}\right)^\omega & \text{, if } 0 \leq k < n_0 \leq n \\ \lambda_{k,n} \leq \left(\frac{k+1}{n+1}\right)^\omega & \text{, if } n_0 \leq k \leq n \end{array}$$

The coefficients given in (6) are then as follows:

- If $0 \leq n < n_0$, then $A_n = \left(1 - \frac{\alpha}{2\beta}\right)^n$, $B_{p,n} = 0$, and

$$\begin{array}{ll} C_{p,n} = \frac{d_q^2}{2\beta} \max_{1 \leq k \leq n} \left(1 - \frac{\alpha}{2\beta}\right)^{n-k} \leq \frac{d_q^2}{2\beta} & \forall p \in [1,2] \\ C_{p,n} = \frac{d_q^2}{2\beta} \left(\sum_{k=1}^n \left(1 - \frac{\alpha}{2\beta}\right)^{(n-k)p_2}\right)^{1/p_2} \leq \frac{d_q^2}{2\beta}\left(1 - \left(1 - \frac{\alpha}{2\beta}\right)^{p_2}\right)^{-1/p_2} \leq \frac{d_q^2}{\alpha} & \forall p > 2 \end{array}$$

  where the last inequality holds since $\frac{d_q^2}{2\beta}\left(1 - \left(1 - \frac{\alpha}{2\beta}\right)^{p_2}\right)^{-1/p_2}$ is monotonically decreasing for $p_2 \in [1, \infty)$ and takes value d_q^2/α at $p_2 = 1$.

- If $n \geq n_0$, then $A_n \leq \left(1-\frac{\alpha}{2\beta}\right)^{n_0-1}\left(\frac{n_0}{n+1}\right)^{\omega}$, and $B_{p,n}$ is given by

$$B_{1,n} \leq \frac{2\gamma d_q}{\alpha}\max_{n_0\leq k\leq n}\frac{(k+1)^{\omega}-k^{\omega}}{(n+1)^{\omega}} \leq \frac{2\gamma d_q}{\alpha}\left(1-\left(\frac{n}{n+1}\right)^{\omega}\right) \leq \frac{2\gamma d_q}{\alpha}\frac{\omega}{n+1}$$
$$B_{p,n} \leq \frac{2\gamma d_q}{\alpha}\left(\sum_{k=n_0}^{n}\left(\frac{(k+1)^{\omega}-k^{\omega}}{(n+1)^{\omega}}\right)^{p_1}\right)^{1/p_1} \leq \frac{2\gamma d_q}{\alpha}\frac{\omega}{n+1}\left(1+\frac{n+1}{1+p_1(\omega-1)}\right)^{1/p_1} \quad \forall p>1$$

where the second last inequality holds since

$$\sum_{k=n_0}^{n}\left(\left(\frac{k+1}{n+1}\right)^{\omega}-\left(\frac{k}{n+1}\right)^{\omega}\right)^{p_1} \leq \sum_{k=n_0}^{n}\left(\frac{\omega}{n+1}\left(\frac{k+1}{n+1}\right)^{\omega-1}\right)^{p_1}$$
$$\leq \left(\frac{\omega}{n+1}\right)^{p_1}\left(1+\sum_{k=n_0}^{n-1}\left(\frac{k+1}{n+1}\right)^{p_1(\omega-1)}\right) \leq \left(\frac{\omega}{n+1}\right)^{p_1}\left(1+(n+1)\int_{\frac{n_0+1}{n+1}}^{1}t^{p_1(\omega-1)}dt\right)$$
$$\leq \left(\frac{\omega}{n+1}\right)^{p_1}\left(1+\frac{n+1}{1+p_1(\omega-1)}\right).$$

For $1 \leq p \leq 2$, $C_{p,n}$ is given by

$$C_{p,n} \leq \left[\frac{d_q^2}{2\beta}\max_{1\leq k<n_0}\left(1-\frac{\alpha}{2\beta}\right)^{n_0-k-1}\left(\frac{n_0}{n+1}\right)^{\omega}\right] \vee \left[\frac{2\beta d_q^2}{\alpha^2}\max_{n_0\leq k\leq n}\left(1-\left(\frac{k}{k+1}\right)^{\omega}\right)^2\left(\frac{k+1}{n+1}\right)^{\omega}\right]$$
$$\leq \frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega} \vee \frac{2\beta d_q^2}{\alpha^2}\max_{n_0\leq k\leq n}\left(\frac{\omega}{k+1}\right)^2\left(\frac{k+1}{n+1}\right)^{\omega}$$
$$= \begin{cases} \frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega} \vee \frac{2\beta d_q^2\omega^2}{\alpha^2}\frac{1}{(n_0+1)^{2-\omega}(n+1)^{\omega}} & \text{, if } 1\leq\omega\leq 2 \\ \frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega} \vee \frac{2\beta d_q^2\omega^2}{\alpha^2}\frac{1}{(n+1)^2} & \text{, if } \omega>2 \end{cases}$$

For $p > 2$, $C_{p,n}$ is given by

$$C_{p,n}^{p_2} \leq \sum_{k=1}^{n_0-1}\left(\frac{d_q^2}{2\beta}\left(1-\frac{\alpha}{2\beta}\right)^{n_0-k-1}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \sum_{k=n_0}^{n}\left(\frac{2\beta d_q^2}{\alpha^2}\left(1-\left(\frac{k}{k+1}\right)^{\omega}\right)^2\left(\frac{k+1}{n+1}\right)^{\omega}\right)^{p_2}$$
$$\leq \left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2}\frac{1}{1-\left(1-\frac{\alpha}{2\beta}\right)^{p_2}} + \left(\frac{2\beta d_q^2}{\alpha^2}\right)^{p_2}\sum_{k=n_0}^{n}\left(\left(\frac{\omega}{k+1}\right)^2\left(\frac{k+1}{n+1}\right)^{\omega}\right)^{p_2}$$
$$\leq \frac{2\beta}{\alpha}\left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \left(\frac{2\beta d_q^2}{\alpha^2}\frac{\omega^2}{(n+1)^{\omega}}\right)^{p_2}\sum_{k=n_0}^{n}(k+1)^{p_2(\omega-2)}.$$

To analyze the second term, let us use the notation $\tilde{\omega} = p_2(\omega-2)$ and note that

$$\begin{aligned} &\textstyle\sum_{k=n_0}^{n}(k+1)^{\tilde{\omega}} \leq \int_{n_0}^{n+1}t^{-1}dt = \log\frac{n+1}{n_0} && \text{, if } \tilde{\omega}=-1 \\ &\textstyle\sum_{k=n_0}^{n}(k+1)^{\tilde{\omega}} \leq \int_{n_0}^{n+1}t^{\tilde{\omega}}dt = \frac{(n+1)^{\tilde{\omega}+1}-n_0^{\tilde{\omega}+1}}{\tilde{\omega}+1} && \text{, if } \tilde{\omega}\in(-\infty,-1)\cup(-1,0] \\ &\textstyle\sum_{k=n_0}^{n}(k+1)^{\tilde{\omega}} \leq (n+1)^{\tilde{\omega}} + \int_{n_0+1}^{n+1}t^{\tilde{\omega}}dt && \\ &\qquad\qquad = (n+1)^{\tilde{\omega}} + \frac{(n+1)^{\tilde{\omega}+1}-(n_0+1)^{\tilde{\omega}+1}}{\tilde{\omega}+1} && \text{, if } \tilde{\omega}>0 \end{aligned}$$

Therefore,

$$
\begin{aligned}
C_{p,n}^{p_2} &\le \frac{2\beta}{\alpha}\left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \left(\frac{2\beta d_q^2}{\alpha^2}\frac{\omega^2}{(n+1)^{\omega}}\right)^{p_2}\frac{n_0^{p_2(\omega-2)+1}}{|p_2(\omega-2)+1|} && \text{, if } p_2(\omega-2) < -1\\
C_{p,n}^{p_2} &\le \frac{2\beta}{\alpha}\left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \left(\frac{2\beta d_q^2}{\alpha^2}\frac{\omega^2}{(n+1)^{\omega}}\right)^{p_2}\log\frac{n+1}{n_0} && \text{, if } p_2(\omega-2) = -1\\
C_{p,n}^{p_2} &\le \frac{2\beta}{\alpha}\left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \left(\frac{2\beta d_q^2}{\alpha^2}\frac{\omega^2}{(n+1)^{2}}\right)^{p_2}\frac{n+1}{p_2(\omega-2)+1} && \text{, if } p_2(\omega-2) \in (-1,0]\\
C_{p,n}^{p_2} &\le \frac{2\beta}{\alpha}\left(\frac{d_q^2}{2\beta}\left(\frac{n_0}{n+1}\right)^{\omega}\right)^{p_2} + \left(\frac{2\beta d_q^2}{\alpha^2}\frac{\omega^2}{(n+1)^{2}}\right)^{p_2}\left(\frac{n+1}{p_2(\omega-2)+1}+1\right) && \text{, if } p_2(\omega-2) > 0
\end{aligned}
$$

To sum up,

$$
\begin{array}{lll}
A_n \in O\left(n^{-\omega}\right) & \text{, if } p \ge 1, & \omega \ge 1\\
B_{p,n} \in O\left(d_q n^{-1/p}\right) & \text{, if } p \ge 1, & \omega \ge 1\\
C_{p,n} \in O\left(d_q^2 n^{-(\omega\wedge 2)}\right) & \text{, if } 1 \le p \le 2, & \omega \ge 1\\
C_{p,n} \in O\left(d_q^2 n^{-\omega}\right) & \text{, if } p > 2, & 1 \le \omega < 1+2/p\\
C_{p,n} \in O\left(d_q^2 n^{-\omega}(\log n)^{1-2/p}\right) & \text{, if } p > 2, & \omega = 1+2/p\\
C_{p,n} \in O\left(d_q^2 n^{-1-2/p}\right) & \text{, if } p > 2, & \omega > 1+2/p
\end{array}
$$

For example, for the case of $p = 1$ and $\omega = 2$, we have

$$A_n \in O\left(n^{-2}\right),\ \ B_{1,n} \in O\left(n^{-1}\right),\ \ C_{1,n} \in O\left(n^{-2}\right)$$

Therefore,

$$
\begin{aligned}
f(\mathbf{x}_n) - f(\mathbf{x}^*) &\le A_n(f(\mathbf{x}_0) - f(\mathbf{x}^*)) + B_{1,n}\|\mathbf{c}_n\|_1 + C_{1,n}\|\mathbf{c}_n\|_1^2\\
&\in O\left(\frac{f(\mathbf{x}_0) - f(\mathbf{x}^*)}{n^2} + \frac{\|\mathbf{c}_n\|_1}{n} + \frac{\|\mathbf{c}_n\|_1^2}{n^2}\right).
\end{aligned} \tag{9}
$$

A.3. Proof of Proposition 11

As stated in Proposition 11 the parameters given in (6) in Lemma 10 yield the tightest coefficients to which (5) holds true for all systems of inequalities of the form of (4). In this section, we prove Proposition 11.

Let $\mathbf{u}_n$ and $\mathbf{v}_n$ be as in the proof of Lemma.10.

- Suppose $\tilde{A} < A_n$. One may take $y_k = \prod_{i=1}^{k}\alpha_i$ and $c_k = 0$ for each k, so that both (4) and (7) are satisfied.

- Suppose $\tilde{B} < B_{p,n}$. It follows by Hölder's inequality that for any $r > 0$, there exists $\mathbf{c}_n = (c_1, ..., c_n) \in \mathbb{R}_{\ge 0}^n$ so that $\|\mathbf{c}_n\|_p = r$ and $\langle \mathbf{u}_n, \mathbf{c}_n\rangle = \|\mathbf{u}_n\|_{p_1} r$. Take non-negative sequence $(y_i)_{i=0}^{\infty}$ so that (4) holds with inequalities replaced by equalities, then (recall that $B_{p,n} = \|\mathbf{u}_n\|_{p_1}$ and $\mathbf{v}_n \ge \mathbf{0}$)

  $$y_n = A_n y_0 + \langle \mathbf{u}_n, \mathbf{c}_n\rangle + \langle \mathbf{v}_n, \mathbf{d}_n\rangle \ge A_n y_0 + B_{p,n} r$$

  where $\mathbf{d}_n = (c_1^2, ..., c_n^2)$. Then (7) holds by taking $r > 0$ and $y_0 > 0$ so that

  $$\tilde{C}r < \frac{1}{2}(B_{p,n} - \tilde{B}),\ \ (\tilde{A} - A_n)y_0 < \frac{1}{2}(B_{p,n} - \tilde{B})r,$$

- Suppose $\tilde{C} < C_{p,n}$. For any $r > 0$, we can construct $\mathbf{c}_n = (c_1, ..., c_n) \in \mathbb{R}^n_{\geq 0}$ so that $\|\mathbf{c}_n\|_p = r$ and $\langle \mathbf{v}_n, \mathbf{d}_n \rangle = C_{p,n} r^2$ (here $\mathbf{d}_n = (c_1^2, ..., c_n^2)$) as follows:
  - If $p > 2$, by Hölder's inequality, there exists $\mathbf{d}_n = (d_1, ..., d_m) \in \mathbb{R}^n_{\geq 0}$ so that $\|\mathbf{d}_n\|_{p/2} = r^2$ and $\langle \mathbf{v}_n, \mathbf{d}_n \rangle = \|\mathbf{v}_n\|_{p_2} r^2 = C_{p,n} r^2$. Take $c_k = \sqrt{d_k}$.
  - If $1 \leq p \leq 2$, take
  $$c_k = \begin{cases} r & \text{, if } k = k_0 \\ 0 & \text{, otherwise} \end{cases}$$
  where $k_0 \in \operatorname{argmax}_{1 \leq k \leq n} \beta_k \prod_{i=k+1}^n \alpha_i$, then $\langle \mathbf{v}_n, \mathbf{d}_n \rangle = \|\mathbf{v}_n\|_\infty r^2 = C_{p,n} r^2$.

  Take non-negative sequence $(y_i)_{i=0}^\infty$ so that (4) holds with inequalities replaced by equalities, then (recall that $\mathbf{u}_n \geq \mathbf{0}$)
  $$y_n = A_n y_0 + \langle \mathbf{u}_n, \mathbf{c}_n \rangle + \langle \mathbf{v}_n, \mathbf{d}_n \rangle \geq A_n y_0 + C_{p,n} r^2.$$
  Then (7) holds by taking $r > 0$ and $y_0 > 0$ so that
  $$\tilde{B} < \frac{1}{2}(C_{p,n} - \tilde{C}) r, \;\; (\tilde{A} - A_n) y_0 < \frac{1}{2}(C_{p,n} - \tilde{C}) r^2.$$

Appendix B. Numerical Evaluations

In this section, we provide experimental results on the worst-case optimization error of the GD with adversarial corruptions.

B.1. Experimental Setup

GD Loop: The experiments use a GD loop based on the GD update rule given in (1). In particular, provided an initial point $\mathbf{x}_0$, objective function f and its gradient $\mathbf{g}$, the number n of iterations and the corruption sequence $(\boldsymbol{\xi}_k)_{k=1}^n$, we iteratively obtain $(\mathbf{x}_k)_{k=1}^n$ based on (1). Recall $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$. Let
$$R_n\left((\boldsymbol{\xi}_k)_{k=1}^n\right) = f(\mathbf{x}_n) - f(\mathbf{x}^*)$$
denote the optimization error corresponding to a particular choice of corruption sequence $(\boldsymbol{\xi}_k)_{k=1}^n$. In contrast, the worst-case optimization error r_n defined in (3) corresponds to the maximum value of $R_n\left((\boldsymbol{\xi}_k)_{k=1}^n\right)$, for the worst case choice of $(\boldsymbol{\xi}_k)_{k=1}^n$ subject to the budget constraint. The learning rate sequence $(\eta_k)_{k=1}^n$ is selected according to Corollary 7.

Adversary: The adversary chooses the corruption sequence $(\boldsymbol{\xi}_k)_{k=1}^n$ in a way that maximizes $R_n\left((\boldsymbol{\xi}_k)_{k=1}^n\right)$, and satisfies the budget constraint with $p = 1$. We formulate this as a constrained optimization problem, and solve it using the *Sequential Least Squares Programming (SLSQP)* method from `scipy.optimize` package in the `SciPy` Python library. This module takes $R_n : \mathbb{R}^n \to \mathbb{R}$ and a random initialization $(\hat{\boldsymbol{\xi}}_k)_{k=1}^n$ of the corruption sequence as the inputs and returns a numerical approximation of the worst case corruptions $(\hat{\boldsymbol{\xi}}_k^*)_{k=1}^n$. We then use $R_n((\hat{\boldsymbol{\xi}}_k^*)_{k=1}^n)$ as a numerical approximation of the minimax error r_n.

Budget: We choose $b(n) \sim sn^\zeta$ for various values of scale $s \in \mathbb{R}_{>0}$ and exponent $\zeta \in [0, 1]$ parameters. In particular, rather than a smooth budget $b(n) = sn^\zeta$, we create $b(n)$ randomly, in a way that it scales with sn^ζ. In order to do this, we choose $b(n) = s(\sum_{k=1}^n \varepsilon_k)^\zeta$, where ε_k are independent uniform random variables in $[0.5, 1.5]$.

More Experimental Details: In our experiments, the domain $\mathcal{X}$ is selected as $[-2, 2]$ interval. Since $\mathcal{X}$ is one dimensional, we use x to denote $\mathbf{x}$, and use ξ_k to denote $\boldsymbol{\xi}_k$ in the subsequent context. Without loss of generality we choose $x^* = 0$. The parameters α, β, and γ are obtained by $\alpha = \min_{x\in\mathcal{X}} \frac{|f'(x)|^2}{f(x)}$, $\beta = \max_{x\in\mathcal{X}} |f''(x)|$, $\gamma = \max_{x\in\mathcal{X}} |f'(x)|$. For GD Loop, we set $x_0 = 2$. Furthermore we restrict the iterations to stay within $\mathcal{X}$. For adversary, the parameters of the *SLSQP* method are given as $ftol = 1e-12$, $eps = 1e-14$, $maxiter = 10000$. Our experiments are run in our internal cluster with 10 CPU servers managed by IBM Spectrum LSF. Each CPU server has 32 CPU cores and 256G RAM. The operating system of these servers are Ubuntu 20.04 LTS. Other implementation details can be found in our source code, which will be released after the review period.

B.2. Assumptions on Objective Functions

A line of research has investigated other conditions which permit establishing analytical results while conforming to practical considerations. Examples include: essential strong convexity [12], weak strong convexity [14], restricted secant inequality [18], and quadratic growth [4]. An interesting condition which is weaker and more general than all mentioned above is the Polyak-Łojasiewicz (PL) condition, which does not even require convexity.

In this experiment, we consider various objective functions: a quadratic objective function referred to as f_{Q} and a strongly convex one referred to as f_{SC}. In addition, we consider two other functions satisfying the PL condition referred to as $f_{\text{PL},1}$ and $f_{\text{PL},2}$; non of them convex; the former is locally convex around x^* and the latter is not. The expression of these objective functions and their plots are given in Table 2 and Figure 1, respectively.

Function Type	Symbol	Function Instance
Quadratic	f_{Q}	$f(x) = x^2$
Strongly Convex	f_{SC}	$f(x) = x^2 - \cos(x) + 1$
Locally Convex Near Optimum	$f_{\text{PL},1}$	$f(x) = x^2 + 0.5 * \sin^2(2x)$
Non-Convex Near optimum	$f_{\text{PL},2}$	$f(x) = x^2 + 2.1 * \sin(\lvert x\rvert^{-1}) * \lvert x\rvert^4$

Table 2: Functions used in the experiments. Note that all functions satisfy the PL condition

B.3. Convergence of the Worst-case Optimization Error

Figure 2 shows the worst-case optimization error of GD for the four objective functions described above. The corruption budget is set to $b(n) \sim \sqrt{n}$. The optimization errors reported in the figures are average over three Monte Carlo runs with different random initialization $(\hat{\xi}_k)_{k=1}^n$ of the corruption sequence for the adversary (the *SLSQP* method mentioned above). The theoretical bounds proven in Corollary 7 are also plotted for comparison. In all cases, the theoretical bounds hold true.

B.4. The Effect of Corruption Budget

Figure 3 (a) and (b) show the numerical results on the worst-case optimization error for various corruption budgets by varying their scale and exponent parameters. The corresponding theoretical bounds from Corollary 7 are also plotted for comparison. All the experiments are consistent with the theoretical findings. As expected, optimization error is larger for a greater corruption budget.

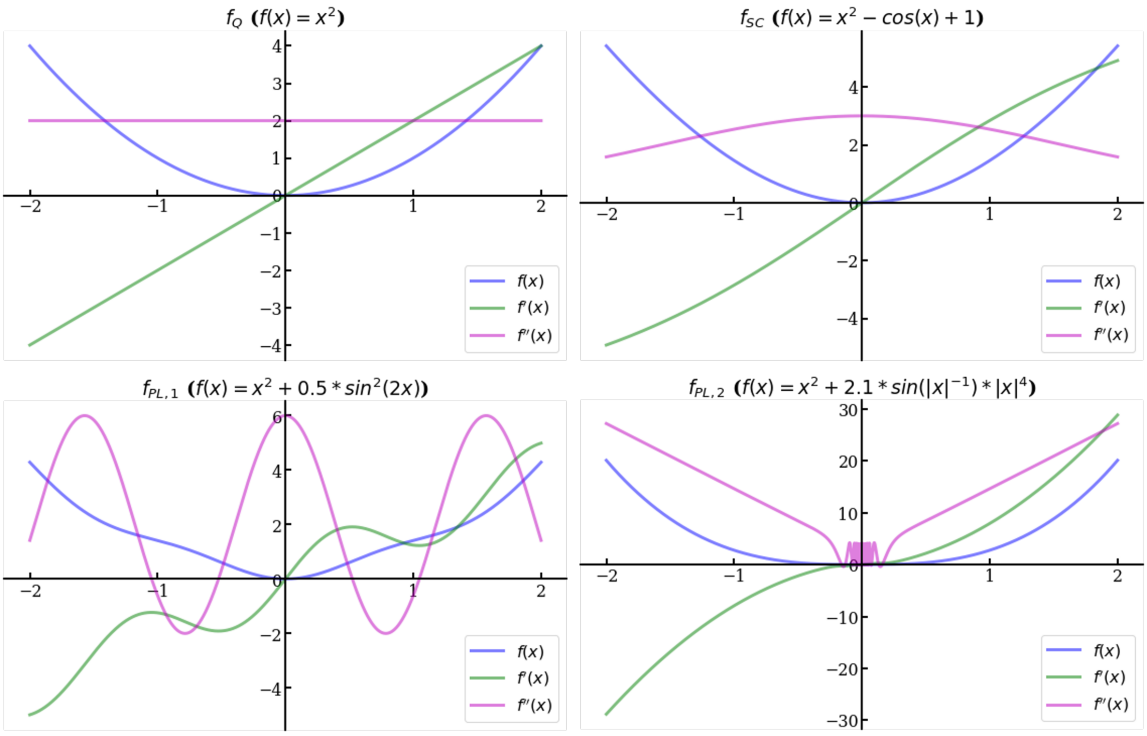

Figure 1: Functions used in the experiments, as well as their first and second order derivatives.

Eventually for a linear budget $b(n) \sim n$ the optimization error does not vanish. We used $f_{\text{PL},2}$ as the objective function for this experiment.

B.5. Adversarial Corruptions

In all previous experiments, we used *SLSQP* method from `scipy.optimize` package [17] (which is under the liberal BSD license) to find a numerical approximation of the worst case corruption sequence. Let us refer to this adversary as $\hat{\mathbf{A}}^*$. In this experiment, we compare the optimization error using $\hat{\mathbf{A}}^*$ versus other adversaries which exploit the available budget in a greedy way. In particular, let $b_{\text{used}}(n) = \sum_{k=1}^{n} c(k)$ denote the amount of budget used up to time n included. Then, the available budget at time k is $b(k) - b_{\text{used}}(k-1)$. The adversary $\mathbf{A}_1$ selects $\xi_k^{\mathbf{A}_1} = (b(k) - b_{\text{used}}(k-1))\,\text{sgn}(-\tilde{\mathbf{g}}(x_{k-1}))$. That is a greedy way to push the GD iterates as far as possible from the minimum, while respecting the budget. We also consider a different adversary, referred to as $\mathbf{A}_2$, which selects $\xi_k^{\mathbf{A}_2} = \xi_k^{\mathbf{A}_1}\varepsilon_k$, where ε_k are independent uniform random variables in $[0.5, 1.5]$. Note that $\mathbf{A}_2$ may violate the budget constraint. It however respects the budget in expectation. Furthermore, we compare these adversaries with the case without adversary, denoted as $\mathbf{A}_0$. Figure 3(c) shows a comparison between the performances of these adversaries; $\hat{\mathbf{A}}^*$ leads to a significantly higher worst-case optimization error compared to $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_0$. Besides, $\mathbf{A}_0$ leads to the lowest worse-case optimization error since it has no adversary. This experiment is run using the objective function $f_{\text{PL},2}$.

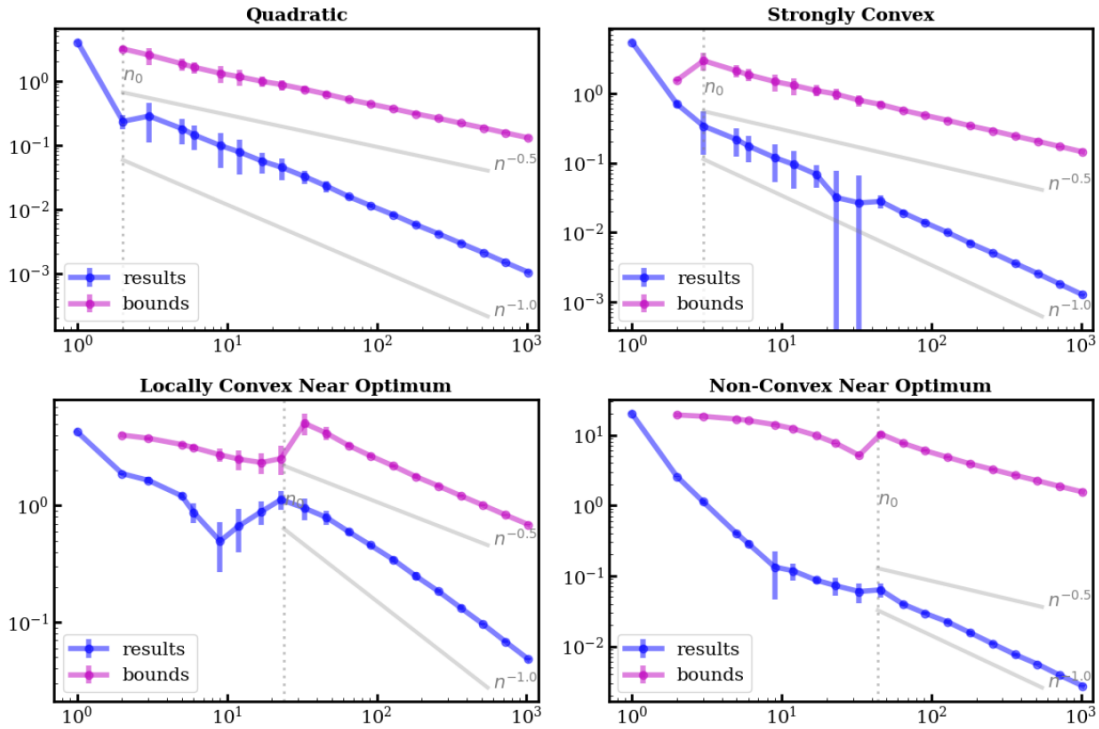

Figure 2: Numerical results on the worst-case optimization error against the number of GD iterations, for various objective functions in logarithmic scale. The x-axis is the number of GD iterations, and the y-axis is the worse-case minimization error. Theoretical bounds on the optimization error from Corollary 7 are also plotted for comparison. The dotted vertical line shows the value of n_0. Grey lines show n^{-1} and $n^{\frac{-1}{2}}$ for comparison. Each error bar shown at each point is the twice standard deviation.

B.6. The Effect of Learning Rate Sequence

In this experiment, we compare the learning rate sequence given in Corollary 7, denoted as $(\eta_k^*)_{k=1}^n$, with other learning rate sequences: $\left(\eta_k^{(1)} = \frac{1}{k}\right)_{k=1}^n$, $\left(\eta_k^{(2)} = \frac{1}{k^2}\right)_{k=1}^n$, and $\left(\eta_k^{(.5)} = \frac{1}{k^{0.5}}\right)_{k=1}^n$. The first one is based on Corollary 7, the others are the similar learning rate sequences but with varying power in their denominators. Figure 3(d) shows a comparison between these learning rate sequences. The worst-case optimization errors of η_k^* and $\eta_k^{(1)}$ decrease steadily, while the errors of $\eta_k^{(2)}$ and $\eta_k^{(.5)}$ stop decreasing after a certain number of GD iterations. This experiment demonstrates that our theorem provides a guideline on learning rate sequences which make GD robust against adversarial corruption.

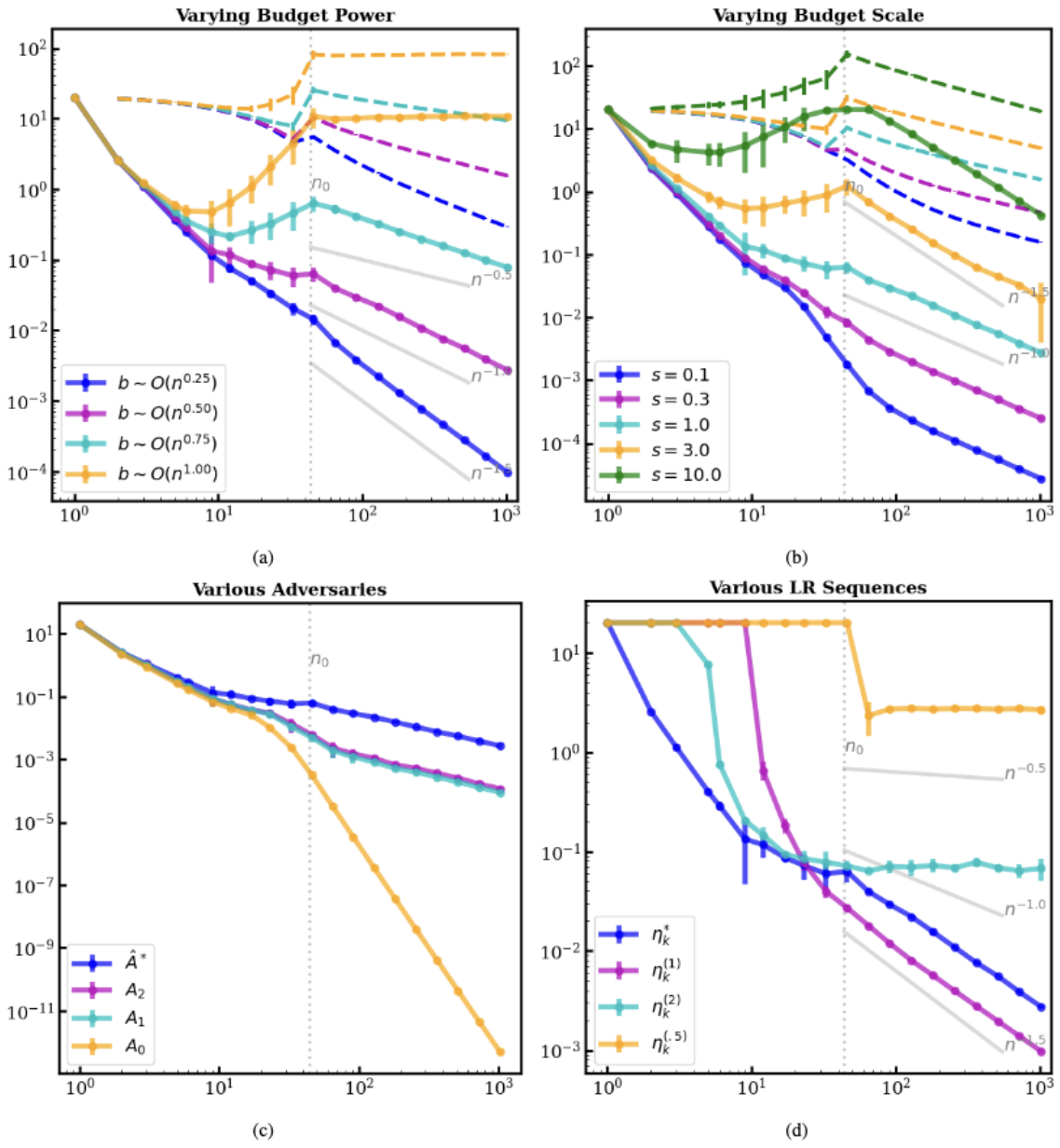

Figure 3: (a) and (b) are numerical results on the worst-case optimization error (solid lines) against the number of GD iterations for various values of the corruption budget in logarithmic scale. The cases of $b(n) \sim n^{\zeta}$ with $\zeta = 0.25, 0.5, 0.75$ and 1 are shown on the left panel. The cases of $b(n) \sim s\sqrt{n}$ with $s = 0.1, 0.3, 1, 3$ and 10 are shown on the right panel. Theoretical bounds from Corollary 7 are plotted (dashed lines) for comparison. This experiment uses $f_{\mathrm{PL},2}$ as the objective function. The dotted vertical line shows the value of n_0. Grey lines show n^{-1}, $n^{-1/2}$ and $n^{-3/2}$ for comparison. (c) and (d) are optimization error against the number of GD iteration for various adversaries and various learning rate sequences in logarithmic scale (see text for description).

B.7. Discussion

In these experiments, we compare the theoretical bounds with the results of numerical experiments on a few example objective functions satisfying the PL condition; including quadratic, strongly convex and non-convex ones. We show how the theoretical and empirical optimization errors are affected by the budget and compare the adversarial corruption with other types of gradient perturbations and other types of learning rate sequences.

However; these experiments indicate that the order of bounds may need to be be improved, at least based on the cases studied here. Furthermore, we use the SLSQP method implemented in the scipy.optimize.minimize package to simulate the adversarial corruption. We only have an approximation of the worst-case corruptions in our experiments, and that is likely to be the reason for the gap between the theoretical upper bound and the results on optimization error in the experiments. The analytical result, however, is a theoretical upper bound and may not realize with all problem instances.