Robustness and scalability under heavy tails, without strong convexity

Matthew J. Holland
Institute of Scientific and Industrial Research
Osaka University

Abstract

Real-world data is laden with outlying values. The challenge for machine learning is that the learner typically has no prior knowledge of whether the feedback it receives (losses, gradients, etc.) will be heavy-tailed or not. In this work, we study a simple, cost-efficient algorithmic strategy that can be leveraged when both losses and gradients can be heavy-tailed. The core technique introduces a simple robust validation sub-routine, which is used to boost the confidence of inexpensive gradient-based sub-processes. Compared with recent robust gradient descent methods from the literature, dimension dependence (both risk bounds and cost) is substantially improved, without relying upon strong convexity or expensive per-step robustification. We also empirically show that the proposed procedure cannot simply be replaced with naive cross-validation.

1 INTRODUCTION

Uncertainty is inherent in real world physical and social systems. This implies that machine learning methods, driven by data generated from these systems, are inherently uncertain. Coupled with our lack of knowledge regarding the mechanisms underlying these systems, it is not possible to provide exact, certain statements regarding algorithm performance. This uncertainty is manifested clearly in the "risk minimization" formulation of learning problems: over some set of candidates $\mathcal{W} \subseteq \mathbb{R}^d$, the *risk* is $R_{\mathrm{P}}(w) := \mathbf{E}_{\mathrm{P}}\, L(w; Z)$, namely the expected value of a loss $L : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}_+$ evaluated at w, where $Z \sim \mathrm{P}$ denotes our random data, taking values in a set $\mathcal{Z}$. In the context of machine learning (by empirical risk minimization), this notion was popularized in the statistical community by the work of Vapnik (1982), and in the computer science community by the work of Haussler (1992). A learning algorithm will have access to n data points sampled from P, denoted $Z_1, \ldots, Z_n$. Write $(Z_1, \ldots, Z_n) \mapsto \widehat{w}_n$ to denote the output of an arbitrary learning algorithm. The usual starting point for analyzing algorithm performance is the *estimation error* $R_{\mathrm{P}}(\widehat{w}_n) - R^*_{\mathrm{P}}$, where $R^*_{\mathrm{P}} := \inf\{R_{\mathrm{P}}(w) : w \in \mathcal{W}\}$, or more precisely, the distribution of this error. Since we never know much about the underlying data-generating process, typically all we can assume is that P belongs to some class $\mathcal{P}$ of probability measures on $\mathcal{Z}$, and typical guarantees are given in the form of $\mathbf{P}\left\{R_{\mathrm{P}}(\widehat{w}_n) - R^*_{\mathrm{P}} > \varepsilon\left(n, \delta, \mathrm{P}, \mathcal{W}\right)\right\} \leq \delta$, over a class $\mathrm{P} \in \mathcal{P}$. In words, the procedure yielding $\widehat{w}_n$ will obtain ε-good performance with $(1-\delta)$-high confidence over the draw of the sample. Citing Holland (2021), in order to have meaningful performance guarantees, the following properties are important.

1. **Transparency:** can we actually compute the output $\widehat{w}_n$ that we study in theory?
2. **Strength:** what form do bounds on $\varepsilon(n, \delta, \mathrm{P}, \mathcal{W})$ take? How rich is the class $\mathcal{P}$?
3. **Scalability:** how do computational costs scale with the above-mentioned factors?

Balancing these three points is critical to developing guarantees for *algorithms that will actually be used in practice*.

Our problem setting This work considers the setup of *potentially heavy-tailed* data, and in contrast with the recent work of Holland (2021), we only assume a *convex* loss, rather than strong convexity. All the learner can know is that for some $m < \infty$,

$$\mathcal{P} \subseteq \left\{\mathrm{P} : \sup_{w \in \mathcal{W}} \mathbf{E}_{\mathrm{P}}\, |L(w; Z)|^m < \infty\right\}, \qquad (1)$$

where typically $m = 2$. Thus, it is unknown whether the losses (or partial derivatives, etc.) are congenial in

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

a sub-Gaussian sense (where (1) holds for all m), or heavy-tailed in the sense that all higher-order moments could be infinite or undefined. The goal then comes down to obtaining the strongest possible guarantees for a tractable learning algorithm, given (1). We next review the related technical literature, and give an overview of our contributions.

2 CONTEXT AND CONTRIBUTIONS

Challenges without strong convexity When one is lucky enough to have a μ-strongly convex risk R_{P}, using a very simple basic idea, a wide range of *distance*-based algorithmic strategies are available (Minsker, 2015; Hsu and Sabato, 2016; Holland, 2021). For example, say we construct k candidates $\widehat{w}^{(1)}, \ldots, \widehat{w}^{(k)}$, and we know that with high probability, a majority of the candidates are ε-good in terms of the risk R_{P}. Since R_{P} is unknown, we can never know which candidates are the ε-good ones. However, this barrier can be circumvented by utilizing the fact that μ-strong convexity of R_{P} implies that any ε-good candidate must be at least $\sqrt{2\varepsilon/\mu}$-close to w^*, the minimizer of R_{P} on $\mathcal{W}$. It follows that on the "good event" in which the majority of candidates are ε-good, it is sufficient to simply "follow the majority." This can be done in various ways, but in the end all such procedures comes down to computing and comparing distances $\|w - \widehat{w}^{(j)}\|$ for all $j \in [k]$. This can be done without knowing which of the $\widehat{w}^{(j)}$ are ε-good, which makes the problem tractable.

Unfortunately, μ-strong convexity is a luxury that is often unavailable. In particular for high-dimensional settings, it is common for the strong convexity parameter μ to shrink rapidly as d grows, making $1/\mu$-dependent error bounds vacuous (Bach and Moulines, 2014). Algorithmically, if strong convexity cannot be guaranteed, then the distance-based strategy just described will fail, since for any particular minimizer w^*, it is perfectly plausible to have a ε-good candidate which is arbitrarily far from w^*. Even when we assume λ_1-smoothness of the risk, all we can say is that ε-badness implies $\sqrt{2\varepsilon/\lambda_1}$-farness from all minimizers; the converse need not hold. The traditional approach: if from sample $\boldsymbol{Z}_n$ we obtain independent candidates $\widehat{w}^{(1)}, \ldots, \widehat{w}^{(k)}$, and we have a second sample $\boldsymbol{Z}'_n = (Z'_1, \ldots, Z'_n)$ available for "validation," then classical procedures return

$$\widehat{w}_n = \widehat{w}^{(\star)}, \text{ where } \star \in \arg\min \left\{ \frac{1}{n} \sum_{i=1}^{n} L(\widehat{w}^{(j)}; Z'_i) : j = 1, \ldots, k \right\}. \quad (2)$$

This technique of confidence boosting for bounded losses is well-known; see (Kearns and Vazirani, 1994, Ch. 4.2) or (Shalev-Shwartz et al., 2010, Thm. 26). Under exp-concave distributions, Mehta (2016) also recently made use of this technique. Problems arise, however, when the losses can be potentially heavy-tailed. The quality of the validated final candidate is only as good as the precision of the risk estimate, and the empirical risk is well-known to be sub-optimal under potentially heavy-tailed data (Devroye et al., 2016).

Limitations of existing procedures To begin, empirical risk minimization (ERM) is the cornerstone of classical learning theory, which studies the statistical properties of any minimizer of the empirical risk, i.e., the sample mean of the losses. Concrete implementations of ERM just require minimizing a finite sum, and thus are computationally quite congenial, and scale well, taken at face value. However, formal guarantees for ERM-based procedures are limited; the empirical mean is known to be sensitive to outliers, and this sensitivity appears in weak formal guarantees. Concretely, under potentially heavy-tailed losses, the empirical mean is sub-optimal in that it cannot guarantee sub-Gaussian deviation bounds. Put roughly, it cannot guarantee error bounds better than those which scale as $\Omega(1/\delta)$; see Catoni (2012) and Devroye et al. (2016) for more details. Furthermore, since ERM leaves the method of implementation completely abstract, this leaves open a large conceptual gap. Feldman (2017) showed lucidly how there exist both "good" and "bad" ERM solutions; the problem with transparency is that we can never know whether any particular ERM candidate is one of the good ones or not. In contrast, starting with seminal work by Brownlees et al. (2015), a recent line of work has led to new statistical learning procedures to address the weak guarantees and lack of robustness of ERM. The basic idea is simply to minimize a different estimator of the risk, for example median-of-means estimators (Minsker, 2018) or M-estimators (Brownlees et al., 2015). Under weak moment bounds like (1), their minimizers enjoy $\mathcal{O}(1/\sqrt{n})$ rates with $\mathcal{O}(\log(\delta^{-1}))$ dependence on the confidence. This provides a significant improvement in terms of the strength of guarantees over ERM, but unfortunately the issue of transparency remains. Like ERM, the algorithmic side of the problem is left abstract here, and in general may even be a much more difficult computational task. As such, the gap between formal guarantees and the guarantees that hold for any given output of the algorithm may be even more severe than in the case of ERM.

Furthermore, several new families of algorithms have been designed in the past few years to tackle the potentially heavy-tailed setting using a tractable procedure. Such algorithms may naturally be called *robust gradient descent* (RGD), since the core update takes the

same form as steepest-descent applied directly to the risk, i.e., $\widehat{w}_{t+1} \leftarrow \widehat{w}_t - \alpha \nabla R_{\mathrm{P}}(w_t)$, except that ∇R_{P} is replaced with an estimator $\widehat{G}_n(w) \approx \nabla R_{\mathrm{P}}(w)$ which has deviations with near-optimal confidence intervals under potentially heavy-tailed data (Chen et al., 2017a; Prasad et al., 2018; Lecué et al., 2018; Holland, 2019; Holland and Ikeda, 2019a,b). Under strong convexity, all these proposals enjoy excess risk bounds with optimal dependence on n and $1/\delta$ under potentially heavy-tailed data, and can be implemented as-is. Unfortunately, instead of a simple one-dimensional robust mean estimate as in Brownlees et al. (2015), all RGD methods rely on sub-routines that work in d-dimensions. This makes the procedures much more expensive computationally for "big" learning tasks, and leads to an undesirable dependence on the ambient dimension d in the statistical guarantees as well. Moreover, when strong convexity is not available, the propagation of statistical error over time for RGD methods becomes much worse, leading to bounds that are extremely sensitive to mis-specified smoothness parameters, step sizes, and total number of iterations.

Our contributions Considering the above limitations of existing techniques, notably the lack of scalability and weak formal guarantees available for RGD methods under weak convexity, here we study a different algorithmic approach to the problem. Our approach has equal generality, and the hope is to achieve as-good or better dependence on n, d, and $1/\delta$ under potentially heavy-tailed data (losses and/or partial derivatives), without strong convexity, and in provably less time for larger problems. The main technique that we investigate is a natural robustification of classical confidence boosting (2), applied to traditional stochastic gradient descent routines run in parallel, though we note that the basic argument can be easily generalized to other optimization strategies (e.g., accelerated methods, adaptive methods, quasi-Newton techniques, etc.). Our main contributions:

- We study a general-purpose robust learning procedure (Algorithm 1), obtaining sharp risk bounds (Theorem 1) that improve on the poor dependence of RGD methods on the dimension and number of iterations under weak convexity (see Table 1).
- The archetype given in Algorithm 1 is concrete, easy to implement as-is, and trivial to run in parallel. All else equal, for high-dimensional learning tasks we can expect to obtain a result as good or better than existing serial RGD methods in far less time.
- Empirically, we study the robustness of our proposed procedure and relevant competitors to various perturbations in the experimental conditions, simulating a lack of prior knowledge about noise levels and convexity. The proposed procedure can easily match benchmark RGD methods in less time, over a variety of test settings. We also verify that a naive cross-validation heuristic does not achieve the same level of performance.

3 THEORETICAL ANALYSIS

3.1 Preliminaries

Notation First we establish some basic notation, and organize numerous technical assumptions in one place for ease of reference. For any positive integer k, write $[k] := \{1, \ldots, k\}$. For any index $\mathcal{I} \subseteq [n]$, write $\boldsymbol{Z}_{\mathcal{I}} := (Z_i)_{i \in \mathcal{I}}$, defined analogously for independent copy $\boldsymbol{Z}'_{\mathcal{I}}$. To keep the notation simple, in the special case of $\mathcal{I} = [n]$, we write $\boldsymbol{Z}_n := \boldsymbol{Z}_{[n]} = (Z_1, \ldots, Z_n)$. We shall use $\mathbf{P}$ as a generic symbol to denote computing probability; in most cases this will be the product measure induced by the sample $\boldsymbol{Z}_n$ or $\boldsymbol{Z}'_n$. For any function $f : \mathbb{R}^d \to \mathbb{R}$, denote by $\partial f(u)$ the sub-differential of f evaluated at u. Variance of the loss is denoted by $\sigma^2_{\mathrm{P}}(w) := \mathrm{var}_{\mathrm{P}}\, L(w; Z) = \mathbf{E}_{\mathrm{P}}(L(w; Z) - R_{\mathrm{P}}(w))^2$ for each $w \in \mathcal{W}$. Denote by $I\{\texttt{event}\}$ the indicator function that returns a value of 1 when `event` is true, and 0 otherwise.

Running assumptions The two key running assumptions that we make are related to independence and convexity. First, we assume that all the observed data are independent, i.e., the random variables Z_i and Z'_i taken over all $i \in [n]$ are independent copies of $Z \sim \mathrm{P}$. Second, for each $z \in \mathcal{Z}$, we assume the map $w \mapsto L(w; z)$ is a real-valued convex function over $\mathbb{R}^d$, and that the parameter set $\mathcal{W} \subseteq \mathbb{R}^d$ is non-empty, convex, and compact, with diameter $\Delta := \sup\{\|u - v\| : u, v \in \mathcal{W}\}$. All results derived in the next sub-section will be for an arbitrary choice of $\mathrm{P} \in \mathcal{P}$, where $\mathcal{P}$ satisfies (1) with $m = 2$. We say a function f is λ_1-*smooth* if its gradient is λ_1-Lipschitz continuous. Finally, to make formal statements technically simpler, we assume that $R_{\mathrm{P}}(\cdot)$ achieves its minimum on the interior of $\mathcal{W}$.

3.2 Error bounds when both losses and gradients can be heavy-tailed

Recalling the challenges described in section 2, we consider a straightforward robustification of the classical validation-based approach using robust mean estimators in a sub-routine. The full procedure is summarized in Algorithm 1. First we outline the key points of the procedure, then give a general-purpose excess risk bound in Theorem 1.

Method	Error	Cost
`RV-SGDAve`	$\mathcal{O}\left(\sqrt{\frac{\log(\delta^{-1})}{n}}\left(\sigma_{L,\mathrm{P}}+\sigma_{G,\mathrm{P}}\right)\right)+\mathcal{O}\left(\frac{\lambda_1\log(\delta^{-1})}{n}\right)$	$\mathcal{O}(dn\log(\delta^{-1}))$
RGD (Chen et al., 2017b)	$\mathcal{O}\left((1+\lambda_1\alpha)^T\sqrt{\frac{d(\sigma^2_{G,\mathrm{P}}\log(d\delta^{-1})+\log(n))}{nT}}\right)$	$\mathcal{O}\left(Tdn\log(\delta^{-1})\right)$

Table 1: High-probability error bounds and computational cost estimates for `RV-SGDAve` (Algorithm 1), compared with modern RGD methods, *without* assuming strong convexity. Error denotes confidence intervals for $R_{\mathrm{P}}(\widehat{w}_n)-R^*_{\mathrm{P}}$ with $\widehat{w}$ being the output of each procedure after T steps (noting `RV-SGDAve` has $T=n$ by definition). Detailed explanation is in supplementary materials.

Algorithm 1 Divide-and-conquer with robust validation; $\texttt{RV-SGDAve}\left[\boldsymbol{Z}_n, \boldsymbol{Z}'_n, \widehat{w}_0; k\right]$.

inputs: samples $\boldsymbol{Z}_n$ and $\boldsymbol{Z}'_n$, initial value $\widehat{w}_0 \in \mathcal{W}$, parameter $1 \leq k \leq n$.

Split $\bigcup_{j=1}^{k} \mathcal{I}_j = [n]$, with $|\mathcal{I}_j| \geq \lfloor n/k \rfloor$, and $\mathcal{I}_j \cap \mathcal{I}_l = \emptyset$ when $j \neq l$.

For each $j \in [k]$, set $\overline{w}^{(j)}$ to the mean of the sequence $\texttt{SGD}[\widehat{w}_0; \boldsymbol{Z}_{\mathcal{I}_j}, \mathcal{W}]$.

Compute $\star = \arg\min_{j\in[k]} \texttt{Valid}\left[\overline{w}^{(j)}; \boldsymbol{Z}'_n\right]$.

return: $\widehat{w}_{\texttt{RV}} = \overline{w}^{(\star)}$.

Core procedure Viewed at a high level, Algorithm 1 is comprised of three extremely simple steps: partition, train, and validate. For our purposes, the key to improving on traditional ERM-style boosting techniques is to ensure the validation step is done with sufficient precision, even when the losses can be heavy-tailed. To achieve this, we shall require that there exist a constant $c > 0$ which does not depend on the distribution P, such that for any choice of confidence level $\delta \in (0,1)$ and large enough n, the sub-routine `Valid` satisfies

$$\left|\texttt{Valid}\left[w; \boldsymbol{Z}'_n\right] - R_{\mathrm{P}}(w)\right| \leq c\sqrt{\frac{(1+\log(\delta^{-1}))\sigma^2_{\mathrm{P}}(w)}{n}} \tag{3}$$

with probability no less than $1-\delta$. Recall that we are denoting $\sigma^2_{\mathrm{P}}(w) := \mathrm{var}_{\mathrm{P}}\, L(w;Z)$, thus the only requirement on the class of data distributions is finite variance, readily allowing for both heavy-tailed losses and gradients. The training step can be done in any number of ways; for concreteness and clarity of the results, we elect to use a simple stochastic gradient descent sub-process. Unpacking the notation from Algorithm 1, the basic update used is traditional projected (sub-)gradient descent, with update denoted by

$$\texttt{SGD}\left[w; Z, \alpha, \mathcal{W}\right] := \Pi_{\mathcal{W}}\left(w - \alpha\, G(w;Z)\right). \tag{4}$$

Here $\alpha \geq 0$ denotes a step-size parameter, $\Pi_{\mathcal{W}}$ denotes projection to $\mathcal{W}$ with respect to the ℓ_2 norm, and the standard assumption is that the random vector $G(w;Z)$ satisfies $\mathbf{E}_{\mathrm{P}}\, G(w;Z) \in \partial R_{\mathrm{P}}(w)$, for each $w \in \mathcal{W}$. Then for arbitrary sequence $(Z_1, \ldots, Z_m)$, we define

$$\texttt{SGD}\left[\widehat{w}_0; (Z_1,\ldots,Z_m), \mathcal{W}\right] := \left\{\texttt{SGD}\left[\widehat{w}_t; Z_{t+1}, \alpha_t, \mathcal{W}\right] : t = 0,1,\ldots,m-1\right\},$$

noting we have suppressed step-sizes from the notation for readability. Replacing the generic sequence $(Z_1,\ldots,Z_m)$ here with $\boldsymbol{Z}_{\mathcal{I}_j}$ for each $j \in [k]$ yields the iterate sequences used in Algorithm 1, with each $\overline{w}^{(j)}$ being simply the arithmetic (vector) mean of the iterates. As all the data we work with are independent copies of $Z \sim \mathrm{P}$, the order in which we take the indices from each $\mathcal{I}_j$ does not matter. Under weak assumptions on the underlying loss distribution, the output $\widehat{w}_{\texttt{RV}}$ of this algorithm enjoys strong excess risk bounds, as the following theorem shows.

Theorem 1. *Let R_{P} be λ_1-smooth in the ℓ_2 norm, $\mathbf{E}_{\mathrm{P}}\left\|G(w;Z) - \nabla R_{\mathrm{P}}(w)\right\|_2^2 \leq \sigma^2_{G,\mathrm{P}} < \infty$, and $\sigma^2_{\mathrm{P}}(w) \leq \sigma^2_{L,\mathrm{P}} < \infty$ for all $w \in \mathcal{W}$. Run Algorithm 1 with sub-routine `Valid` satisfying (3), given a total sample size $n \geq 2k$ split into $\boldsymbol{Z}_{n/2}$ and $\boldsymbol{Z}'_{n/2}$, and `SGD` sub-processes using step sizes $\alpha_t = 1/(\lambda_1 + (1/a))$, where $a = \Delta/\sqrt{n\sigma^2_{G,\mathrm{P}}/2k}$. If we set $k = \lceil \log(2\lceil\log(\delta^{-1})\rceil\delta^{-1})\rceil$, then for any confidence parameter $0 < \delta \leq 1/3$, we have*

$$R_{\mathrm{P}}(\widehat{w}_{\texttt{RV}}) - R^*_{\mathrm{P}} \leq 2c\sqrt{\frac{2(1+\log(2\lceil\log(\delta^{-1})\rceil\delta^{-1}))\sigma^2_{L,\mathrm{P}}}{n}} + 3\left(\frac{k\Delta^2\lambda_1}{n} + \sqrt{\frac{2k\Delta^2\sigma^2_{G,\mathrm{P}}}{n}}\right)$$

with probability no less than $1-3\delta$.

Algorithm 2 Catoni type M-estimate.

inputs: sample $\{u_1, \ldots, u_n\}$, parameters $\sigma > 0$ and $0 < \delta < 1$.

Set $q^2 = \dfrac{2\sigma^2 \log(2\delta^{-1})}{n - 2\log(2\delta^{-1})}$ and $s^2 = \dfrac{n(\sigma^2 + q^2)}{2\log(2\delta^{-1})}$.

return: $\displaystyle\operatorname*{arg\,min}_{\theta \in \mathbb{R}} \sum_{i=1}^{n} \rho\left(\frac{u_i - \theta}{s}\right)$.

We shall look at concrete implementations of `Valid` in section 4, but as an initial example, setting `Valid` to be a properly scaled M-estimator (sub-routine given in Algorithm 2) satisfies (3) with $c \leq 2$ whenever $n \geq 4\log(\delta^{-1})$. Additional examples and supporting lemmas are given in the supplementary materials.

Proof sketch Here we give an overview of the proof of Theorem 1. We have data sequences $\boldsymbol{Z}_n$ and $\boldsymbol{Z}'_n$. The former is used to obtain independent candidates $\overline{w}^{(1)}, \ldots, \overline{w}^{(k)}$, and the latter is used to select among these candidates. As mentioned earlier, distance-based strategies (Minsker, 2015; Hsu and Sabato, 2016) require that the *majority* of these candidates are ε-good, in order to ensure that points near the majority coincide with ε-good points. In our present setup, where strong convexity is not available, we are taking a very different approach. Now we only require that *at least one* of the candidates is ε-good. Making this explicit,

$$\mathcal{E}_1(\varepsilon; k) := \bigcup_{j=1}^{k} \left\{ R_{\mathrm{P}}(\overline{w}^{(j)}) - R^*_{\mathrm{P}} \leq \varepsilon \right\} \tag{5}$$

is our first event of interest. Note that *if* for each $j \in [k]$ we have an upper bound $\varepsilon_{\mathrm{P}}(\cdot)$ depending on the sample size for the sub-process outputs $\overline{w}^{(j)}$ such that

$$\mathbf{E}\left[R_{\mathrm{P}}(\overline{w}^{(j)}) - R^*_{\mathrm{P}} \right] \leq \varepsilon_{\mathrm{P}}(\lfloor n/k \rfloor),$$

where expectation is taken over the subset indexed by $\mathcal{I}_j$, then using Markov's inequality and taking a union bound, it follows that setting $\varepsilon = \mathrm{e}\,\varepsilon_{\mathrm{P}}$, we have $\mathbf{P}\,\mathcal{E}_1(\mathrm{e}\,\varepsilon_{\mathrm{P}}; k) \geq 1 - \mathrm{e}^{-k}$. Asking for one ε-good candidate is a much weaker requirement than asking for the majority to be ε-good, but we must pay the price in a different form, as we require that `Valid` provide a good estimate of the true risk for *all* of the k candidates. In particular, writing $b_{\mathrm{P}}(n, \delta)$ for a confidence interval to be specified shortly, this is the following event:

$$\mathcal{E}_2(\delta; k) := \bigcap_{j=1}^{k} \left\{ \left| \texttt{Valid}\left[\overline{w}^{(j)}; \boldsymbol{Z}'_n \right] - R_{\mathrm{P}}(\overline{w}^{(j)}) \right| \leq b_{\mathrm{P}}(n, \delta) \right\}.$$

Intuitively, while we only require that at least one of the k candidates be good, we must reliably know which is *best* at the available precision, which requires paying the price of the intersection defining $\mathcal{E}_2(\delta; k)$. Recalling the requirement (3), if we condition on $\boldsymbol{Z}_n$, the candidates $\overline{w}^{(1)}, \ldots, \overline{w}^{(k)}$ become non-random elements of $\mathcal{W}$, which means that setting $b_{\mathrm{P}}(n, \delta) = c\sqrt{(1 + \log(\delta^{-1}))\sigma^2_{L,\mathrm{P}}/n}$, a union bound gives us $\mathbf{P}\left(\mathcal{E}_2(\delta; k); \boldsymbol{Z}_n\right) \geq 1 - k\delta$. This inequality holds as-is for any realization of $\boldsymbol{Z}_n$, so we can thus integrate to obtain

$$\mathbf{P}\,\mathcal{E}_2(\delta; k) = \int \mathbf{P}\left(\mathcal{E}_2(\delta; k); \boldsymbol{Z}_n\right) \mathrm{P}(\mathrm{d}\boldsymbol{Z}_n) \geq 1 - k\delta.$$

The good event of interest then has probability

$$\mathbf{P}\left[\mathcal{E}_1\left(\mathrm{e}\,\varepsilon_{\mathrm{P}}\left(\left\lfloor \frac{n}{k} \right\rfloor \right); k \right) \cap \mathcal{E}_2(\delta; k) \right] \geq 1 - \mathrm{e}^{-k} - k\delta.$$

On this good event, we know that there does exist an ε-good candidate, even though we can never know which it is; call it $\overline{w}_{\textsc{luck}} \in \{\overline{w}^{(1)}, \ldots, \overline{w}^{(k)}\}$. Furthermore, even though this candidate is unknown, since we have $b_{\mathrm{P}}(n, \delta)$-good risk estimates for all k candidates, the choice of $\overline{w}^{(\star)}$, with $\star = \arg\min_{j \in [k]} \texttt{Valid}[\overline{w}^{(j)}; \boldsymbol{Z}'_n]$, cannot be much worse. More precisely, we have

$$\begin{aligned}
&R_{\mathrm{P}}(\overline{w}^{(\star)}) - R^*_{\mathrm{P}} \\
&\quad = R_{\mathrm{P}}(\overline{w}^{(\star)}) - \texttt{Valid}[\overline{w}^{(\star)}] + \texttt{Valid}[\overline{w}^{(\star)}] - R^*_{\mathrm{P}} \\
&\quad \leq R_{\mathrm{P}}(\overline{w}^{(\star)}) - \texttt{Valid}[\overline{w}^{(\star)}] + \texttt{Valid}[\overline{w}_{\textsc{luck}}] - R^*_{\mathrm{P}} \\
&\quad = \left[R_{\mathrm{P}}(\overline{w}^{(\star)}) - \texttt{Valid}[\overline{w}^{(\star)}] \right] + \\
&\qquad [\texttt{Valid}[\overline{w}_{\textsc{luck}}] - R_{\mathrm{P}}(\overline{w}_{\textsc{luck}})] + [R_{\mathrm{P}}(\overline{w}_{\textsc{luck}}) - R^*_{\mathrm{P}}] \\
&\quad \leq 2b_{\mathrm{P}}(n, \delta) + \mathrm{e}\,\varepsilon_{\mathrm{P}}\left(\lfloor n/k \rfloor\right).
\end{aligned}$$

We have effectively proved the following lemma.

Lemma 2 (Boosting the confidence under potentially heavy tails)**.** *Assume we have a learning algorithm* `Learn` *such that for $n \geq 1$ and $\delta \in (0, 1)$, we have*

$$\mathbf{P}\left\{ R_{\mathrm{P}}(\texttt{Learn}[\boldsymbol{Z}_n]) - R^*_{\mathrm{P}} > \frac{\varepsilon_{\mathrm{P}}(n)}{\delta} \right\} \leq \delta.$$

Splitting the data $\boldsymbol{Z}_n$ using sub-indices $\mathcal{I}_1, \ldots, \mathcal{I}_k$, if we set

$$\star = \operatorname*{arg\,min}_{j \in [k]} \texttt{Valid}\left[\texttt{Learn}[\boldsymbol{Z}_{\mathcal{I}_j}]; \boldsymbol{Z}'_n \right],$$

then when `Valid` *satisfies (3), it follows that for any $\delta \in (0, 1)$, we have*

$$\begin{aligned}
&R_{\mathrm{P}}\left(\texttt{Learn}[\boldsymbol{Z}_{\mathcal{I}_\star}]\right) - R^*_{\mathrm{P}} \leq \\
&\qquad \sup_{w \in \mathcal{W}} 2c\sqrt{\frac{(1 + \log(\delta^{-1}))\sigma^2_{\mathrm{P}}(w)}{n}} + \mathrm{e}\,\varepsilon_{\mathrm{P}}\left(\left\lfloor \frac{n}{k} \right\rfloor \right)
\end{aligned}$$

with probability no less than $1 - k\delta - \mathrm{e}^{-k}$.

Note that Lemma 2 here makes no direct requirements on the underlying loss or risk, beyond the need for a variance bound, which appears as $\sigma_{\mathrm{P}}^2(w) \leq \sigma_{L,\mathrm{P}}^2 < \infty$ in the statement of Theorem 1. Indeed, convexity does not even make an appearance. This is in stark contrast with distance-based confidence boosting methods, which rely upon the strong convexity of the risk (Minsker, 2015; Hsu and Sabato, 2016). As such, so long as we can validate in the sense of (3), then Lemma 2 gives us a general-purpose tool from which we can construct algorithms with competitive risk bounds under potentially heavy-tailed data. This can be done for many practical procedures, and the only main step that remains is to clean up the good event probability and specify k to achieve the properties stated in Theorem 1. Letting δ be that given in the theorem statement, first set $\delta_0 = \delta/(2\lceil\log(\delta^{-1})\rceil) < \delta$. Next, set the number of subsets to be

$$k = \lceil\log(1/\delta_0)\rceil = \lceil\log(2\lceil\log(\delta^{-1})\rceil\delta^{-1})\rceil,$$

and note that with this setting of k and δ_0, we have that

$$\begin{aligned}
1 - k\delta_0 &= 1 - \lceil\log(2\lceil\log(\delta^{-1})\rceil\delta^{-1})\rceil \left(\frac{\delta}{2\lceil\log(\delta^{-1})\rceil}\right) \\
&\geq 1 - \left(\frac{\lceil\log(2)\rceil}{\lceil\log(\delta^{-1})\rceil} + \frac{\lceil\log(\log(\delta^{-1}))\rceil}{\lceil\log(\delta^{-1})\rceil} + 1\right)\frac{\delta}{2} \\
&\geq 1 - \left(\frac{3}{2}\right)\delta \\
&\geq 1 - 2\delta.
\end{aligned}$$

The inequalities follow readily via the fact that for arbitrary $c_1, c_2 \geq 0$ we have $\lceil c_1 + c_2\rceil \leq \lceil c_1\rceil + \lceil c_2\rceil$, and that $\lceil\log(2)\rceil/\lceil\log(\delta^{-1})\rceil \leq 1$ for all $\delta \leq 1/2$. As for the exponential term, note that

$$\mathrm{e}^{-k} = \exp\left(-\lceil\log(\delta_0^{-1})\rceil\right) \leq \exp\left(-\log(\delta_0^{-1})\right) = \delta_0 < \delta.$$

It thus immediately follows that the desired good event holds with probability no less than

$$1 - \mathrm{e}^{-k} - k\delta_0 \geq 1 - \delta - 2\delta = 1 - 3\delta.$$

Tying together these basic facts readily allows us to prove Theorem 1 (detailed proof in the supplementary materials).

4 EMPIRICAL ANALYSIS

To study how the theoretical insights obtained in the previous section play out in practice, we carried out a series of tightly controlled numerical tests. The basic experimental design strategy that we employ is to calibrate all the methods (learning algorithms) of interest to achieve good performance under a particular learning setup, and then we systematically modify characteristics of the learning tasks, leaving the methods fixed, to observe how performance changes in both an absolute and relative sense. Viewed from a high level, the main points we address can be categorized as follows:

(E1) How do error trajectories of baseline methods change via robust validation?

(E2) How does relative performance change in high dimensions without strong convexity?

(E3) How do actual computation times compare as n and/or d grow?

(E4) Can robust validation be replaced by cross-validation?

Experimental setup We essentially follow the standard "noisy convex minimization" tests used in the literature to test the robustness of RGD methods (Holland and Ikeda, 2019a). Complete details of the experimental setup are provided in the supplementary materials.[1] Put simply, we provide the learner with random losses of the form $L(w; Z) = (\langle w - w^*, X\rangle + E)^2/2$, where $w^* \in \mathbb{R}^d$ is a pre-defined vector unknown to the learner, X is a d-dimensional random vector, E is zero-mean random noise, and X and E are independent of each other. This approach is advantageous in that we can compute the resulting risk $R_{\mathrm{P}}(w) = \mathbf{E}_{\mathrm{P}}\, L(w; Z)$ exactly, and by modifying the distribution P, we can ensure that even while allowing heavy-tailed losses/gradients, we still satisfy the key technical assumptions of Theorem 1, namely λ_1-smooth R_{P} and gradients with $\sigma_{G,\mathrm{P}}$-bounded variance, plus with the μ-strong convexity of R_{P} is at our control, we can construct many flat directions, and observe behaviour as $\mu \downarrow 0$.

With respect to the different methods being studied, we use a mixture of classical baselines and modern alternatives to compare with our Algorithm 1 based on SGD with and without averaging, denoted `RV-SGD` and `RV-SGDAve` respectively. The sub-routine `Valid` is carried out using a Catoni-type M-estimator (Catoni, 2012). For baselines, we do empirical risk minimization using batch gradient descent (denoted `ERM-GD`) and stochastic gradient descent, both with and without averaging (denoted `SGD` and `SGD-Ave`). Several representative robust gradient descent methods discussed in section 2 are implemented here, including RGD via median-of-means (Chen et al., 2017b; Prasad et al., 2018) (denoted `RGD-MoM`), median-of-means minimization by gradient descent (Lecué et al., 2018) (denoted

[1] Software repository:
https://github.com/feedbackward/sgd-roboost

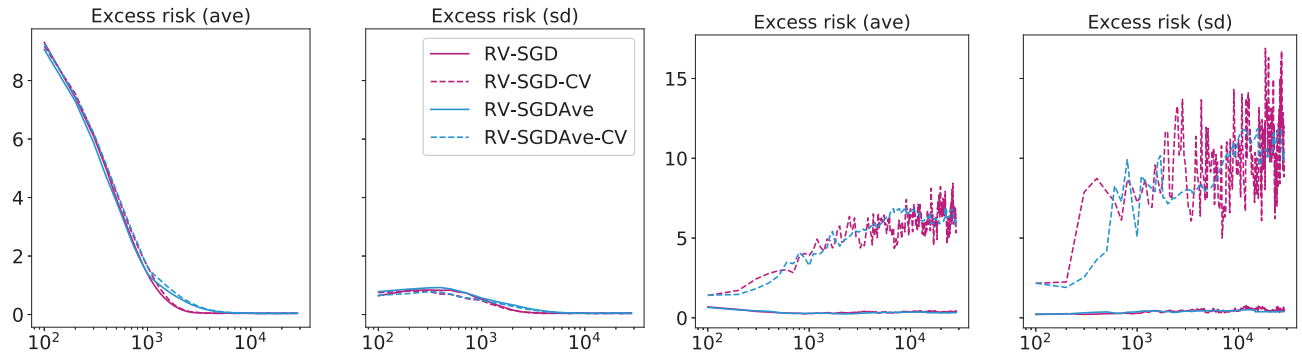

Figure 1: The negative impact of trying to modify Algorithm 1 to use a cross validation heuristic. Left: Normal noise. Right: log-Normal noise.

`RGD-Lec`), and RGD via M-estimation (Holland and Ikeda, 2019a) (denoted `RGD-M`). Finally, we also study a cross validation heuristic (marked by `-CV` suffix), where instead of splitting the sample for validation, we do k-fold cross validation using the full training data set. Essentially, the candidates $\overline{w}^{(j)}$ are computed just as in Algorithm 1, except without the split into $\boldsymbol{Z}_{n/2}$ and $\boldsymbol{Z}'_{n/2}$, and `Valid` is used to evaluate each candidate using the held-out data for each $j \in [k]$. The final candidate is the one for which `Valid` on the held-out data returns the smallest value. This gives the sub-processes double the data for training, but sacrifices the independence of the data used for validation. Detailed settings of each method, including random initialization, are given in the supplementary materials.

Discussion of results Representative results are given in Figures 2–1. Starting with proof-of-concept test results given in Figure 2, we see how even very noisy sub-processes can be ironed out easily using the simple robust validation sub-routine included in Algorithm 1, and that even running the algorithm for much longer than a single pass over the data, risk which is comparable to benchmark RGD methods can be realized at a much smaller cost, with comparable variance across trials, and that this holds under both sub-Gaussian and heavy-tailed data, without any modifications to the procedure being run. A particularly lucid improvement in the cost-performance tradeoff is evident in Figure 3, since near-identical performance can be achieved at a small fraction of the computational cost. Note that under Normal noise, running Algorithm 1 for just a single pass leaves room for improvement performance-wise, but as we saw in the low-dimension case, in practice this can be remedied by taking additional passes over the data. Finally, regarding the question of whether or not Algorithm 1 can be replaced with a naive k-fold cross-validation heuristic, the answer is clear (Figure 1): while the results are comparable under well-behaved data (the Normal noise case here), when heavy tails are a possibility (e.g., the log-Normal case), the naive cross-validation method fails to get even near the performance of Algorithm 1.

5 FUTURE DIRECTIONS

The main take-away from this initial study is that even without strong convexity, under potentially heavy-tailed losses and/or gradients, there exists a computationally efficient procedure which improves upon the formal performance guarantees of modern robust GD techniques, is very competitive in practice, and scales much better to large tasks with many parameters. The basic archetype for such a procedure was illustrated using the concrete Algorithm 1, but naturally this can be extended in many directions, to deal with accelerated, adaptive, or variance-reducing sub-processes, under more general geometries and more challenging constraints applied to $\mathcal{W}$. In particular, if one considers a stochastic mirror descent type of generalization to the proposed algorithm, it would be interesting to compare the robust validation approach taken here with say the truncation-based approach studied recently by Juditsky et al. (2019), and how the performance of the respective methods changes under different constraints on prior knowledge of the underlying data-generating distribution.

Acknowledgements

This work was partially supported by the JSPS KAKENHI Grant Number 19K20342, and the Kayamori Foundation of Information Science Advancement Grant Number K30-XXIII-518.

References

Bach, F. and Moulines, E. (2014). Non-strongly-convex smooth stochastic approximation with convergence

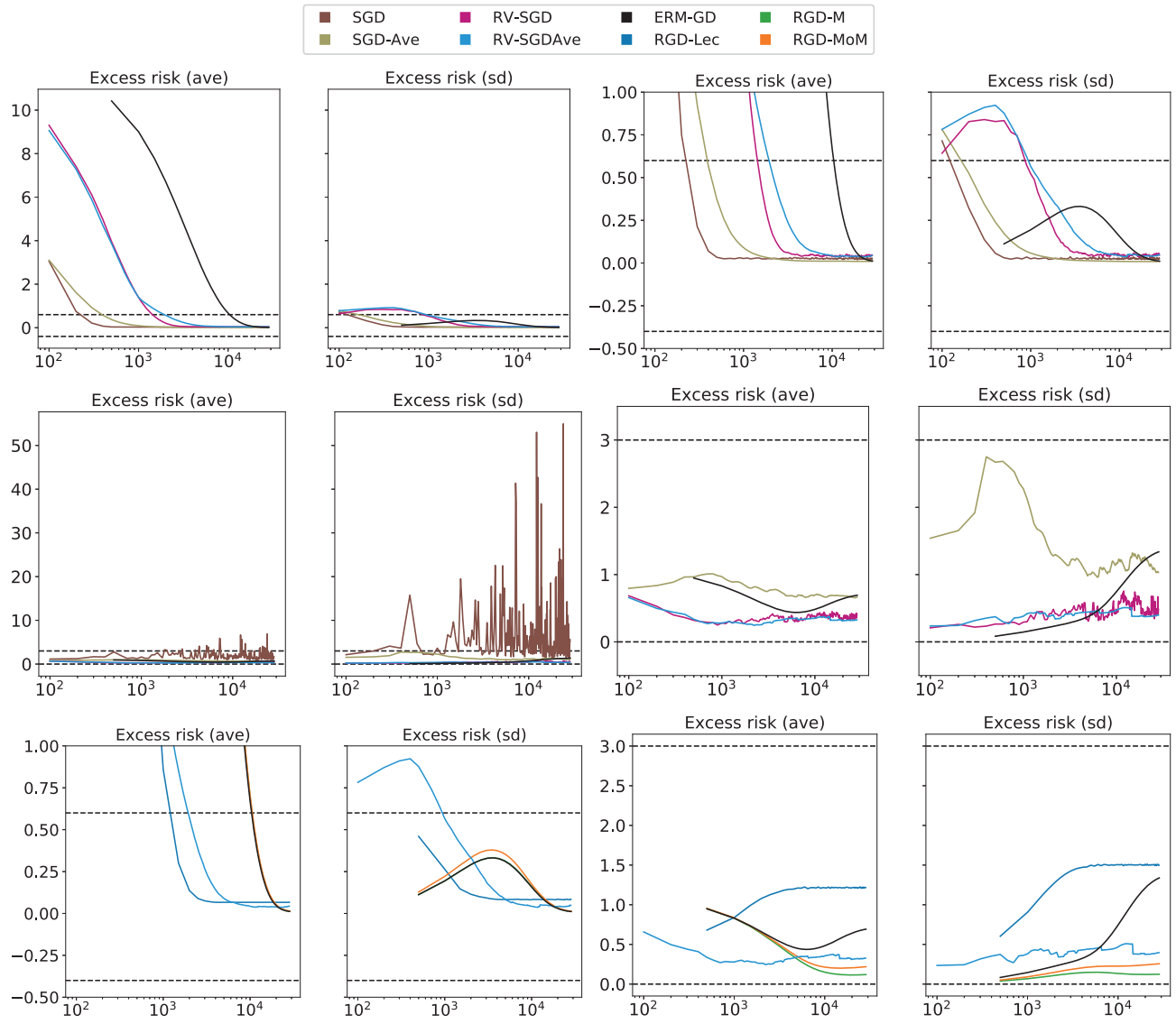

Figure 2: Excess risk statistics (`ave` and `sd`) as a function of cost in gradients (log scale, base 10). Top row: Normal noise. Middle row: log-Normal noise. Right-most plots show a zoomed-in view of left-most plots. Bottom row: comparison with RGD methods, Normal (left) and log-Normal (right).

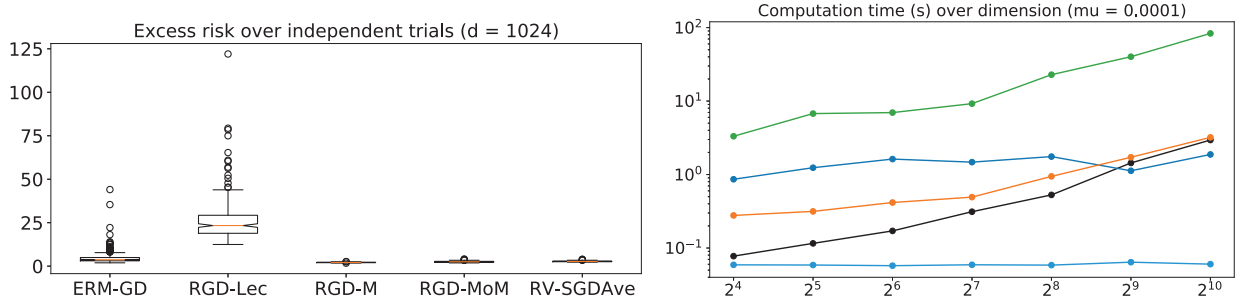

Figure 3: Performance under $\mu = 0.0001$. Left: box-plot of final excess risk for batch methods versus `RV-SGDAve`, with $d = 1024$ under log-Normal noise. Right: median time cost over d.

rate $o(1/n)$. In *Advances in Neural Information Processing Systems 26*, pages 773–781.

Brownlees, C., Joly, E., and Lugosi, G. (2015). Empirical risk minimization for heavy-tailed losses. *Annals of Statistics*, 43(6):2507–2536.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Optimization*, 8(3–4):231–357.

Catoni, O. (2012). Challenging the empirical mean and empirical variance: a deviation study. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 48(4):1148–1185.

Chen, Y., Su, L., and Xu, J. (2017a). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *arXiv preprint arXiv:1705.05491v2*.

Chen, Y., Su, L., and Xu, J. (2017b). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*. ACM.

Cohen, M. B., Lee, Y. T., Miller, G., Pachocki, J., and Sidford, A. (2016). Geometric median in nearly linear time. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, pages 9–21.

Devroye, L., Lerasle, M., Lugosi, G., and Oliveira, R. I. (2016). Sub-gaussian mean estimators. *Annals of Statistics*, 44(6):2695–2725.

Feldman, V. (2017). Generalization of ERM in stochastic convex optimization: The dimension strikes back. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 3576–3584.

Haussler, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150.

Holland, M. J. (2019). Robust descent using smoothed multiplicative noise. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 703–711.

Holland, M. J. (2021). Scaling-up robust gradient descent techniques. In *35th AAAI Conference on Artificial Intelligence (AAAI 2021)*.

Holland, M. J. and Ikeda, K. (2019a). Better generalization with less data using robust gradient descent. In *36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*.

Holland, M. J. and Ikeda, K. (2019b). Efficient learning with robust gradient descent. *Machine Learning*, 108(8):1523–1560.

Hsu, D. and Sabato, S. (2016). Loss minimization and parameter estimation with heavy tails. *Journal of Machine Learning Research*, 17(18):1–40.

Juditsky, A., Nazin, A., Nemirovsky, A., and Tsybakov, A. (2019). Algorithms of robust stochastic optimization based on mirror descent method. *arXiv preprint arXiv:1907.02707v1*.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Lecué, G., Lerasle, M., and Mathieu, T. (2018). Robust classification via MOM minimization. *arXiv preprint arXiv:1808.03106v1*.

Lugosi, G. and Mendelson, S. (2019). Mean estimation and regression under heavy-tailed distributions: A survey. *Foundations of Computational Mathematics*, 19(5):1145–1190.

Mehta, N. A. (2016). Fast rates with high probability in exp-concave statistical learning. *arXiv preprint arXiv:1605.01288*.

Minsker, S. (2015). Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335.

Minsker, S. (2018). Uniform bounds for robust mean estimators. *arXiv preprint arXiv:1812.03523*.

Nemirovsky, A. S. and Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. Wiley-Interscience.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Springer.

Prasad, A., Suggala, A. S., Balakrishnan, S., and Ravikumar, P. (2018). Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*.

Shalev-Shwartz, S., Shamir, O., Srebro, N., and Sridharan, K. (2010). Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11:2635–2670.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics. Springer-Verlag.

6 APPENDIX

The following materials are included in this appendix:

- Additional formal proofs (section 6.1).

- Comparison of error bounds summarized in Table 1 (section 6.2).
- Empirical supplement (section 6.3).

6.1 Additional proofs

The following lemma shows that validation can indeed be done in the desired way, using straightforward computational procedures.

Lemma 3. *The following implementations of* $\texttt{Valid}[w;\cdot]$ *satisfy (3) with sample size* n *and confidence level* $0 < \delta < 1$*, when passed sample* $\{L(w;Z_i') : i \in [n]\}$*.*

- $\texttt{MoM}[\cdot;k']$ *(Algorithm 3), with* $c \leq 2\sqrt{2}\mathrm{e}$*, when* $k' = \lceil \log(\delta^{-1}) \rceil$ *and* $n \geq 2(1+\log(\delta^{-1}))$*.*
- $\texttt{CM}[\cdot;\sigma_{\mathrm{P}}^2(w),\delta]$ *(Algorithm 4), with* $c \leq 2$*, when* $n \geq 4\log(\delta^{-1})$*.*
- $\texttt{LM}[\cdot;\delta]$ *(Algorithm 5), with* $c \leq 9\sqrt{2}$*, when* $n \geq (16/3)\log(8\delta^{-1})$*.*

Proof of Lemma 3. For the median-of-means estimator MoM, see Devroye et al. (2016, Sec. 4.1), or Hsu and Sabato (2016) for a lucid proof. For the M-estimator CM, simply apply Catoni (2012, Prop. 2.4). For the truncated mean estimator, see Lugosi and Mendelson (2019, Thm. 6). □

Proof of Theorem 1. Since most key facts have already been laid out, we just need to fill in a few blanks and connect these facts. To begin, consider the $\varepsilon_{\mathrm{P}}(\cdot)$-bound on Learn in Lemma 2. The special case of Algorithm 1 is just $\texttt{Learn}[\boldsymbol{Z}_n] = \texttt{Average}\,[\texttt{SGD}[\widehat{w}_0;\boldsymbol{Z}_n,\mathcal{W}]]$, namely the simplest form of averaged stochastic gradient descent. Given the assumptions, we are doing averaged SGD under a λ_1-smooth risk, without assuming strong convexity or a Lipschitz loss, and using the step-sizes specified in the hypothesis, a standard argument gives us

$$\varepsilon_{\mathrm{P}}(n) \leq \left(\frac{\Delta^2\lambda_1}{2n} + \sqrt{\frac{\Delta^2\sigma_{G,\mathrm{P}}^2}{n}} \right), \tag{6}$$

where $\sigma_{G,\mathrm{P}}^2$ is as given in the theorem statement. See Theorem 4 in the appendix for a proof of the more general result that implies (6). This can be applied to each sub-process via the correspondence $\overline{w}^{(j)} \leftrightarrow \texttt{Learn}[\boldsymbol{Z}_{\mathcal{I}_j}]$. Note that the output of Algorithm 1 corresponds to $\widehat{w}_{\texttt{RV}} \leftrightarrow \texttt{Learn}[\boldsymbol{Z}_{\mathcal{I}_\star}]$. Thus leveraging Lemma 2 and (6), and bounding $\sigma_{\mathrm{P}}^2(w) \leq \sigma_{L,\mathrm{P}}^2$, we have for any choice of $\delta_0 \in (0,1)$ that

$$R_{\mathrm{P}}(\widehat{w}_{\texttt{RV}}) - R_{\mathrm{P}}^* \leq 2c\sqrt{\frac{(1+\log(\delta_0^{-1}))\sigma_{L,\mathrm{P}}^2}{n}} + \mathrm{e}\left(\frac{k\Delta^2\lambda_1}{2n} + \sqrt{\frac{k\Delta^2\sigma_{G,\mathrm{P}}^2}{n}} \right) \tag{7}$$

with probability no less than $1 - \mathrm{e}^{-k} - k\delta_0$. Note that we are assuming k divides n for simplicity, and using the notation δ_0 to distinguish from δ in the theorem statement. Cleaning up this probability and specifying k has already been described in the main text, and as such, setting the number of subsets to be

$$k = \lceil \log(1/\delta_0) \rceil = \lceil \log(2\lceil \log(\delta^{-1}) \rceil \delta^{-1}) \rceil,$$

and note that with this setting of k and δ_0, we have that the good event of (7) holds probability no less than

$$1 - \mathrm{e}^{-k} - k\delta_0 \geq 1 - \delta - 2\delta = 1 - 3\delta.$$

To conclude, since we have n observations split in half, we must replace n with $n/2$ in (7). Bounding the coefficient $\mathrm{e} \leq 3$ for simplicity yields the desired result. □

In the proof of Theorem 1, one key underlying result we rely on has to do with convergence rates of averaged SGD for smooth objectives. Recall that assuming $f : \mathcal{V} \to \mathbb{R}$ is differentiable, we say that f is *λ-smooth* in norm $\|\cdot\|$ if its gradients are λ-Lipschitz continuous in the same norm, that is

$$\|\nabla f(u) - \nabla f(v)\| \leq \lambda\|u - v\| \tag{8}$$

for all $u, v \in \mathcal{V}$. Nesterov (2004, Thm. 2.1.5) gives many useful characterizations of λ-smoothness. The fact cited directly in the main text is summarized in the following theorem; it can be extracted readily from well-known properties of (stochastic) mirror descent, a family of algorithms dating back to Nemirovsky and Yudin (1983).

Theorem 4 (Convex and smooth case; averaged)**.** *Let* R_{P} *be* λ_1*-smooth in the* ℓ_2 *norm. Furthermore, assume that* $\mathbf{E}_{\mathrm{P}}\|G(w;Z) - \nabla R_{\mathrm{P}}(w)\|_2^2 \leq \sigma_{G,\mathrm{P}}^2 < \infty$ *for all* $w \in \mathcal{W}$*. Run* SGD *(4) with step size* $\alpha_t = 1/(\lambda_1 + (1/c_n))$ *for* n *iterations, setting* $c_n = \Delta/\sqrt{\sigma^2 n}$*, and take the average as*

$$\widehat{w}_{[n]} := \frac{1}{n}\sum_{t=1}^{n} \widehat{w}_{t-1}.$$

We then have with probability no less than $1 - \delta$ *that*

$$R_{\mathrm{P}}(\widehat{w}_{[n]}) - R_{\mathrm{P}}^* \leq \frac{\Delta}{\delta}\left(\frac{\Delta\lambda_1}{2n} + \frac{\sigma_{G,\mathrm{P}}}{\sqrt{n}} \right).$$

Proof of Theorem 4. To begin, we establish some extra terms and notation related to mirror descent. For any differentiable convex function $f : \mathcal{V} \to \mathbb{R}$, define the *Bregman divergence* induced by f as

$$D_f(u, v) := f(u) - f(v) - \langle \nabla f(v), u - v \rangle. \quad (9)$$

In mirror descent, one utilizes Bregman divergences of a particular class of convex functions, often called "mirror maps." Let $\mathcal{W}_0 \subseteq \mathbb{R}^d$ be an open convex set containing including $\mathcal{W}$ within its closure, and also let $\mathcal{W} \cap \mathcal{W}_0 \neq \emptyset$. We denote arbitrary mirror maps on $\mathcal{W}_0$ by $\Phi : \mathcal{W}_0 \to \mathbb{R}$. Strictly speaking, to call Φ a *mirror map* on $\mathcal{W}_0$ it is sufficient if Φ is strictly convex, differentiable, and that its gradient takes on all values (i.e., $\{\nabla \Phi(u) : u \in \mathcal{W}_0\} = \mathbb{R}^d$) and diverges on the boundary of $\mathcal{W}_0$; see Bubeck (2015, Sec. 4) and the references therein for more details. Bregman divergences induced by mirror maps, namely $D_\Phi : \mathcal{W}_0 \to \mathbb{R}$, play an important role in mirror descent when constructing a projection map that takes up between the primal space $\mathcal{W}$, and the space where we can leverage gradient information. The generic mirror descent procedure is as follows. Initializing at arbitrary $\widehat{w}_0 \in \mathcal{W} \cap \mathcal{W}_0$, we update as

$$\widehat{w}_{t+1} = \operatorname*{arg\,min}_{u \in \mathcal{W} \cap \mathcal{W}_0} \left[\alpha_t \langle u, G(\widehat{w}_t; Z_t) \rangle + D_\Phi(u, \widehat{w}_t) \right], \quad (10)$$

where the random gradient vector is such that $\mathbf{E}_{\mathrm{P}}\, G(w; Z) \in \partial R_{\mathrm{P}}(w)$ for all $w \in \mathcal{W}$, just as discussed after equation (4). The following result is useful (Bubeck, 2015, Thm. 6.3):

Lemma 5. *Assume* $\mathbf{E}_{\mathrm{P}} \|G(w; Z) - \nabla R_{\mathrm{P}}(w)\|_*^2 \leq \sigma_{G,\mathrm{P}}^2$ *for all* $w \in \mathcal{W}$*, and that* R_{P} *is* λ_1*-smooth in norm* $\|\cdot\|$*. Write* $r^2 = \sup\{\Phi(w) - \Phi(\widehat{w}_0) : w \in \mathcal{W} \cap \mathcal{W}_0\}$*. Run stochastic mirror descent (10) for* n *iterations, using any mirror map* Φ *that is* 1*-strongly convex on* $\mathcal{W} \cap \mathcal{W}_0$ *in norm* $\|\cdot\|$*, with step sizes* $\alpha_t = 1/(\lambda_1 + 1/c_n)$*, using* $c_n = \sqrt{2r^2/(n\sigma_{G,\mathrm{P}}^2)}$*. Under this setting, we have*

$$\mathbf{E}\left[R_{\mathrm{P}}\left(\frac{1}{n} \sum_{i=1}^n \widehat{w}_i \right) - R_{\mathrm{P}}^* \right] \leq \frac{r^2 \lambda_1}{n} + \sqrt{\frac{2r^2 \sigma_{G,\mathrm{P}}^2}{n}},$$

where expectation is taken with respect to the entire sequence $(Z_1, \ldots, Z_n)$*.*

In order to use Lemma 5, it is sufficient to show that SGD (4) is a special case of (10). Letting $\mathcal{W}_0 = \mathbb{R}^d$, and setting $\Phi(u) = \|u\|_2^2/2$, note that this is a valid mirror map, and strong convexity follows from noting that the Hessian of Φ in this special case is the identity matrix. The resulting Bregman divergence is $D_\Phi(u, v) = \|u - v\|_2^2/2$. Noting that for any $u, w \in \mathcal{W}$ we have

$$\langle G(w; Z), u - w \rangle + \frac{1}{2\alpha}\|u - w\|_2^2 = \frac{1}{2\alpha}\|u - (w - \alpha\, G(w; Z))\|_2^2 - \frac{\alpha}{2}\|G(w; Z)\|_2^2,$$

it follows that the left-hand side over $\mathcal{W}$ is minimized by setting $u = \Pi_{\mathcal{W}}(w - \alpha\, G(w; Z))$. Using this fact, it follows that

$$\begin{aligned} \widehat{w}_{t+1} &= \operatorname*{arg\,min}_{u \in \mathcal{W}} \left[\alpha_t \langle u, G(\widehat{w}_t; Z_t) \rangle + \frac{1}{2}\|u - \widehat{w}_t\|_2^2 \right] \\ &= \operatorname*{arg\,min}_{u \in \mathcal{W}} \left[\langle u - \widehat{w}_t, G(\widehat{w}_t; Z_t) \rangle + \frac{1}{2\alpha_t}\|u - \widehat{w}_t\|_2^2 \right] \\ &= \Pi_{\mathcal{W}}\left(\widehat{w}_t - \alpha_t\, G(\widehat{w}_t; Z_t) \right), \end{aligned}$$

which is precisely the SGD update in (4). Since the dual norm $\|\cdot\|_*$ of the ℓ_2 norm is once again the ℓ_2 norm, all the other assumptions in Lemma 5 clearly align with those in Theorem 4, which follows from a direct application of Markov's inequality to convert bounds in expectation to high-probability confidence intervals, and finally using the fact that $r^2 \leq \Delta/\sqrt{2}$. □

Algorithm 3 Median of means estimate; $\texttt{MoM}[\{u_1, \ldots, u_n\}; k]$.

inputs: sample $\{u_1, \ldots, u_n\}$, parameter $1 \leq k \leq n$.

$\bigcup_{j=1}^k \mathcal{I}_j = [n]$, with $|\mathcal{I}_j| \geq \lfloor n/k \rfloor$, and $\mathcal{I}_j \cap \mathcal{I}_l = \emptyset$ when $j \neq l$.

$\widehat{u}_j = \dfrac{1}{|\mathcal{I}_j|} \displaystyle\sum_{i \in \mathcal{I}_j} u_i$, for each $j \in [k]$.

return: $\mathrm{med}\{\widehat{u}_1, \ldots, \widehat{u}_k\}$.

Algorithm 4 Catoni-type M-estimate; $\texttt{CM}[\{u_1, \ldots, u_n\}; \sigma, \delta]$.

inputs: sample $\{u_1, \ldots, u_n\}$, parameters $\sigma > 0$ and $0 < \delta < 1$.

Set $q^2 = \dfrac{2\sigma^2 \log(2\delta^{-1})}{n - 2\log(2\delta^{-1})}$ and $s^2 = \dfrac{n(\sigma^2 + q^2)}{2\log(2\delta^{-1})}$.

return: $\displaystyle\operatorname*{arg\,min}_{\theta \in \mathbb{R}} \sum_{i=1}^n \rho\left(\frac{u_i - \theta}{s} \right)$.

6.2 Comparison of error bounds (summarized in Table 1)

Recall that in the introduction, we highlighted properties of transparency, strength, and stability as being important to close the gap between formal guarantees

Algorithm 5 Truncated mean estimate; $\texttt{LM}[\{u_1, \dots, u_n\}; \delta]$.

inputs: sample $\{u_1, \dots, u_n\}$, parameter $0 < \delta < 1$.

Split the index $[n] = \mathcal{I}_1 \cup \mathcal{I}_2$, with $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$ and $|\mathcal{I}_1| \geq |\mathcal{I}_2| \geq \lfloor n/2 \rfloor$.

Set $\beta = 32 \log(8\delta^{-1})/(3n)$.

Set a and b to the β- and $(1-\beta)$-level quantiles of $\{u_i : i \in \mathcal{I}_2\}$.

return: $\dfrac{1}{|\mathcal{I}_1|} \displaystyle\sum_{i \in \mathcal{I}_1} u_i I_{\{a \leq u_i \leq b\}}$.

and the performance achieved by the methods we actually are coding. As mentioned in the literature review in section 2, the robust gradient descent algorithms cited are noteworthy in that the procedures have strong (albeit slightly sub-optimal) guarantees for a wide class of distributions, for procedures which can be implemented essentially as-stated in the cited papers, making the guarantees very transparent. Unfortunately, the best results are essentially limited to problems in which the risk R_{P} is μ-strongly convex; all of the cited papers make extensive use of this property in their analysis (Chen et al., 2017b; Holland and Ikeda, 2019a; Prasad et al., 2018). If one is lucky enough to have μ-strong convexity (and λ_1-smoothness), then for any step t, one has

$$\|\widehat{w}_t - \alpha_t \nabla R_{\mathrm{P}}(\widehat{w}_t) - w^*\|^2 \leq \left(1 - \frac{2\alpha_t \mu \lambda_1}{\mu + \lambda_1}\right) \|\widehat{w}_t - w^*\|^2.$$

The only difference between the left-hand side of this inequality and the general-purpose robust GD update studied in the literature is that the true risk gradient is replaced with some estimator $\widehat{G}_n \approx \nabla R_{\mathrm{P}}$. As such, one can easily control $\|\widehat{w}_{t+1} - w^*\|$ using an upper bound that depends on the right-hand side of the above inequality and the statistical estimation error $\|\widehat{G}_n(\widehat{w}_t) - \nabla R_{\mathrm{P}}(\widehat{w}_t)\|$. After say T iterations, one can then readily unfold the recursion and obtain final error bounds that can be given as a sum of an optimization error term depending on the number of iterations T, and a statistical error term depending on the sample size n (e.g., Chen et al. (2017a, Thm. 5), Prasad et al. (2018, Sec. 7), Holland and Ikeda (2019a, Thm. 5)).

Error bounds without strong convexity On the other hand, when one does not have strong convexity, such a technique fails, and one is left having to compare the difference between two sequences, the actual robust GD iterates $(\widehat{w}_t)$, and the ideal sequence (w_t^*) of gradient descent using the true risk gradient, assuming both sequences are initialized at the same point $\widehat{w}_0 = w_0^*$. This point is discussed with analysis by Holland and Ikeda (2019b, Sec. A.3).[2] One can still unfold the recursion without much difficulty, but the propagation of the statistical error becomes much more severe. In the simple case using a fixed step-size of $\alpha > 0$, ignoring non-dominant terms, under the same technical assumptions used in our theoretical analysis, after T steps, the robust RGD procedures can only obtain $(1-\delta)$-high probability bounds of the form

$$R_{\mathrm{P}}(\widehat{w}_T) - R_{\mathrm{P}}^* \lesssim \mathcal{O}\left((1 + \lambda_1 \alpha)^T \sqrt{\frac{d(\sigma_{G,\mathrm{P}}^2 \log(d\delta^{-1}) + \log(n))}{n\,T}}\right),$$

Note that the exponential dependence on T makes the maximum number of iterations one can guarantee extremely sensitive to the values of λ_1 and α.

In contrast, under the same assumptions, Theorem 1 for our Algorithm 1 has no such sensitivity; it achieves the same dependence on n and $1/\delta$ with just one pass over the data. Furthermore, there is no explicit dependence on the number of parameters d, the $\log(n)$ factor is removed, and the dependence on λ_1 is improved exponentially. Since typical RGD procedures do not ever use loss values, the only moment bound requirement they make is via $\sigma_{G,\mathrm{P}}^2$, whereas our procedure has both $\sigma_{G,\mathrm{P}}^2$ and $\sigma_{L,\mathrm{P}}^2$. Other minor tradeoffs exist in the form of an extra $\log(\log(\delta^{-1}))$ factor, and dependence on the diameter Δ in our bounds is linear, whereas the works of Chen et al. (2017a) and Holland and Ikeda (2019a) have logarithmic dependence. Arguably, this is a small price to pay for the improvements that are afforded. These results are shown in the second column of Table 1.

Computational cost Due to the ease of distributed computation and simplicity of the underlying subroutines, Algorithm 1 has significant potential to improve upon existing robust GD methods in terms of computational scalability. Using arithmetic operations as a rough estimate of time complexity, first for Algorithm 1 note that for each subset $\mathcal{I}_j$, we have a fixed number of arithmetic operations that must be done for d coordinates and $|\mathcal{I}_j| \geq \lfloor n/k \rfloor$ iterations. Thus one can obtain each candidate $\widehat{w}^{(j)}$ with $\mathcal{O}(dn/k)$ operations, and this is done for each $j \in [k]$. These computations can trivially be done on independent cores running in parallel. It then just remains to make a single call to `Valid` to conclude the procedure. In this final call,

[2] Their original bounds involve a factor dV, where V is an upper bound on the variance of the partial derivatives of the loss taken over all coordinates. One can easily strengthen their bounds by replacing bounds stated using dV with bounds stated using $\sigma_{G,\mathrm{P}}^2$. Analogous analysis can be done to extend the results of Chen et al. (2017b) to the weak convexity case as well.

one evaluates k candidates at $\mathcal{O}(n)$ data points; this will typically require $\mathcal{O}(dkn)$ operations, plus the cost of the final robust estimate, which will be respectively $\text{cost}(\texttt{MoM}) = \mathcal{O}(k\log(k))$ and $\text{cost}(\texttt{CM}) = \mathcal{O}(n)$ for the cases described in Lemma 3. Adding these costs up, ignoring $\log(k)$ factors, and setting $k \propto \log(\delta^{-1})$ for simplicity yields the cost shown in the third column of Table 1.

In contrast, RGD by median-of-means (Chen et al., 2017b; Prasad et al., 2018) requires $\mathcal{O}(dn/k)$ operations to compute one subset mean, and again assuming the computations for each $\mathcal{I}_j$, $j \in [k]$, are done across k cores in parallel, then if T iterations are done, the total cost is $\mathcal{O}(Tdn/k) + T\,\text{cost}(\texttt{GeoMed})$, since the `GeoMed`-based merging must be done T times. Regarding $\text{cost}(\texttt{GeoMed})$, the geometric median is a convex program, and can efficiently be solved to arbitrary accuracy; Cohen et al. (2016) give an implementation such that the `GeoMed` objective is $(1+\varepsilon)$-good (relative value), with time complexity of $\text{cost}(\texttt{GeoMed}) = \mathcal{O}(dk\log^3(\varepsilon^{-1}))$. For RGD by M-estimation (Holland and Ikeda, 2019a), note that solving for $\widehat{\theta}_j(w)$ can be done readily using a fixed-point update, and in practice the number of iterations is $\mathcal{O}(1)$, fixed independently of n and d, which means $\mathcal{O}(dn)$ operations will be required for each of the T steps. Assuming a standard empirical estimate of the per-coordinate variance is plugged in, this will require an additional $\mathcal{O}(dn)$ arithmetic operations. Adding these costs up yields the estimate shown in Table 1.

6.3 Details of empirical analysis

We essentially follow the standard "noisy convex minimization" tests done by Holland and Ikeda (2019a) to compare the performance of robust gradient descent procedures with traditional ERM minimizers. For simplicity, we start with a risk function that takes a quadratic form $R_{\mathrm{P}}(w) = \langle \Sigma w, w\rangle + \langle w, u\rangle + a$, where $\Sigma \in \mathbb{R}^{d\times d}$, $u \in \mathbb{R}^d$, and $a \in \mathbb{R}$ are constants that depend on the experimental conditions. Now in order to line the experimental setting up with the theory of section 3, the idea is to construct an easily manipulated loss distribution such that the expectation aligns precisely with the quadratic R_{P} just given. To achieve this, one can naturally compute losses of the form $L(w;Z) = (\langle w - w^*, X\rangle + E)^2/2$, where $w^* \in \mathbb{R}^d$ is a pre-defined vector unknown to the learner, X is a d-dimensional random vector, E is zero-mean random noise, and X and E are independent of each other. Note that P in this case corresponds to the joint distribution of X and E, although all the learner sees is the loss value. It is readily confirmed under such a setting we have $\mathbf{E}_{\mathrm{P}}\, L(w;Z) = R_{\mathrm{P}}(w)$ in the quadratic form given above with $\Sigma = \mathbf{E}_{\mathrm{P}}\, XX^{\mathrm{T}}/2$, $u = -2\Sigma w^*$, and $a = \langle \Sigma w^*, w^*\rangle + \mathbf{E}_{\mathrm{P}}\, E^2/2$.

With respect to the different methods being studied, we use a mixture of classical baselines and modern alternatives to compare with our Algorithm 1, denoted `RV-SGDAve`, with `Valid` carried out using a Catoni-type M-estimator (Catoni, 2012). For baselines, we do empirical risk minimization using batch gradient descent (denoted `ERM-GD`) and stochastic gradient descent, both with and without averaging (denoted `SGD` and `SGD-Ave`). Several representative robust gradient descent methods discussed in section 2 are implemented here, including RGD via median-of-means (Chen et al., 2017b; Prasad et al., 2018) (denoted `RGD-MoM`), median-of-means minimization by gradient descent (Lecué et al., 2018) (denoted `RGD-Lec`), and RGD via M-estimation (Holland and Ikeda, 2019a) (denoted `RGD-M`). Everything is implemented in Python (ver. 3.8), chiefly relying upon the `numpy` library (ver. 1.18). The basic idea of these tests is to calibrate and fix the methods to the case of "nice" data characterized by additive Gaussian noise, and then to see how the performance of each method changes as different experimental parameters are modified. For all algorithms that use a k-partition of the data, we have fixed $k = 10$ throughout all tests. Partitioning is done using the `split_array` function in `numpy`, which means each subset gets at least $\lfloor n/k\rfloor$ points. Details regarding step-size settings will be described shortly. Finally, the initial value $\widehat{w}_0$ for all methods is determined randomly, using $\widehat{w}_{0,j} = w^*_j + \text{Uniform}[-c,+c]$ for each coordinate, with $c = 5.0$ unless otherwise specified.

The key performance metric that we look at in the figures to follow is "excess risk," computed as $R_{\mathrm{P}}(\widehat{w}) - R_{\mathrm{P}}(w^*)$, where $\widehat{w}$ is the output of any learning algorithm being studied, and w^* is the pre-fixed minimum described in the previous paragraph. Each experimental setting is characterized by the triplet (P, n, d), which we modify in many different ways to investigate different phenomena. For each setting, we run multiple independent trials, and compute performance statistics based on these trials. For example, when we give the average (denoted `ave`) and standard deviation (denoted `sd`) of excess risk, these statistics are computed over all trials. All box-plots are also computed based on multiple independent trials; the actual number of trials will be described in the subsequent exposition. For convenience, here we list the key contents of our empirical analysis:

(E1) How do error trajectories of baseline methods change via robust validation?

(E2) How does relative performance change in high dimensions without strong convexity?

(E3) How do actual computation times compare as n

and/or d grow?

(E4) Can robust validation be replaced by cross-validation?

(E1) How do error trajectories of baseline methods change via robust validation? Before we look at the impact of R_{P} having weak convexity, we begin with a nascent investigation of the basic workings of the robust validation procedure of interest. We run 100 independent trials for both the Normal and log-Normal settings described previously, with $d = 2$, $n = 500$, and 1-strongly convex R_{P}. Here we let all methods run with a fixed "budget" of $40n\sqrt{d}$, where the "cost" is measured by gradient computations, i.e., cost is incremented by one when $\nabla L(w; Z_i)$ is computed at any w for any $i \in [n]$. Naturally, this means Algorithm 1 will be run for multiple passes over the data, meaning that the behavior after the first pass takes us, strictly speaking, beyond the scope of Theorem 1, a natural point of empirical interest. In Figures 4–5, we show how the baseline stochastic methods change when being passed through a robust validation procedure such as is used in our Algorithm 1. Here `RV-SGDAve` is precisely Algorithm 1, where `RV-SGD` denotes the same procedure *without* averaging the SGD sub-processes. It is natural to choose `RV-SGDAve` as a representative, and in Figure 6, we compare just `RV-SGDAve` against the modern RGD methods.

(E2) How does relative performance change in high dimensions without strong convexity? Next we look at how the competing learning algorithms perform as the number of parameters to determine increases, with R_{P} having very weak convexity in many directions. More precisely, the matrix Σ is diagonal, and half the diagonal elements are no greater than 10^{-4}, implying a tiny upper bound on the strong convexity parameter of R_{P}. Under this setting, we look at how increasing d over the range $2 \leq d \leq 1024$, with fixed sample size $n = 2500$ impacts algorithm performance. We run 250 independent trials, and for each trial record performance achieved by each method once it has spent its budget, again measured in gradient computations. Batch methods are given a large budget of $100n$. In contrast, with the previous experiments, here we only let `RV-SGDAve` (Algorithm 1) take one pass over the data for initialization, and one pass for learning, so a budget of just $2n$. This aligns more precisely with the setting of Theorem 1. Noise distribution settings are as previously introduced. In Figure 7, we give box-plots of the final excess risk achieved by each method for different d sizes.

(E3) How do actual computation times compare as n and/or d grow? While we have given a brief discussion of the computational complexity of different algorithms in section 6.2, it is assuredly worthwhile to actually measure how much time is needed to achieve the performance realized in the results of (E2) above. In particular, we are interested in whether or not even a very simple parallel implementation of the SGD sub-processes used in `RV-SGDAve` could lead to substantial time savings. We look at two types of tests. First, n and d move together, with $n = 4000d$ and $2 \leq d \leq 64$. Second, $n = 2500$ is fixed, and dimension ranges over $2 \leq d \leq 1024$ as in (E2). Budget constraints used for stopping rules are exactly as described in (E2). We run 250 independent trials, and compute the median times for each method. We remark that in comparing the log-Normal versus Normal cases, there is virtually no difference between the computation times for any method, and thus to save space we simply show times for the log-Normal case; these median times for both experimental settings are shown in Figure 8.

(E4) Can robust validation be replaced by cross-validation? Finally, it is natural to ask whether the procedure of Algorithm 1 could be replaced by a heuristic cross-validation procedure that uses all the data for learning, doubling the effective sample size available to each sub-process. More precisely, say that instead of splitting the n-sized sample into $\boldsymbol{Z}_{n/2}$ and $\boldsymbol{Z}'_{n/2}$ as done by `RV-SGD` (Algorithm 1), we simply use a full n-sized sample $\boldsymbol{Z}_n$, partition into k subsets $\mathcal{I}_1, \ldots, \mathcal{I}_k$, obtaining k independent candidates $\overline{w}^{(1)}, \ldots, \overline{w}^{(k)}$, now with double the sample size compared with `RV-SGD`. One might be intuitively inclined to do a cross-validation type of selection, where for each $j \in [k]$, the validation score returned by `Valid` is computed for each $\overline{w}^{(j)}$ using the data Z_i indexed by $i \in [n] \setminus \mathcal{I}_j$, and the winning index $\star$ is selected to be the minimizer of this cross-validation error. Such heuristics break the assumptions used in the theoretical analysis of Algorithm 1, and it is interesting to see how this plays out in practice. Thus, we have re-implemented both `RV-SGD` and `RV-SGDAve` in this fashion, respectively denoted `RV-SGD-CV` and `RV-SGDAve-CV`. Error trajectories for the same experimental setting as (E1) for all these methods are compared in Figure 9.

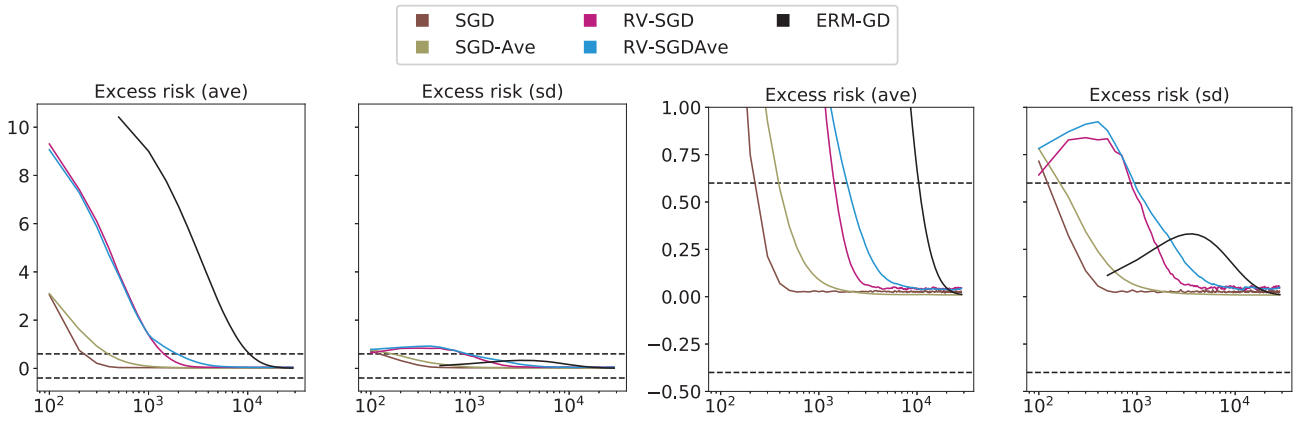

Figure 4: Excess risk statistics as a function of cost in gradients (log scale, base 10). The two right-most plots zoom in on the region between the dashed lines in the two left-most plots.

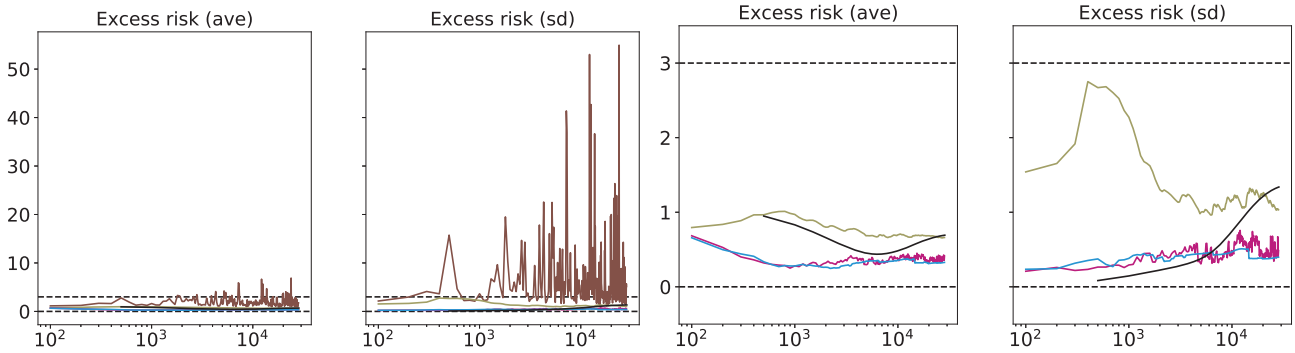

Figure 5: Analogous results to Figure 4, for the case of log-Normal noise.

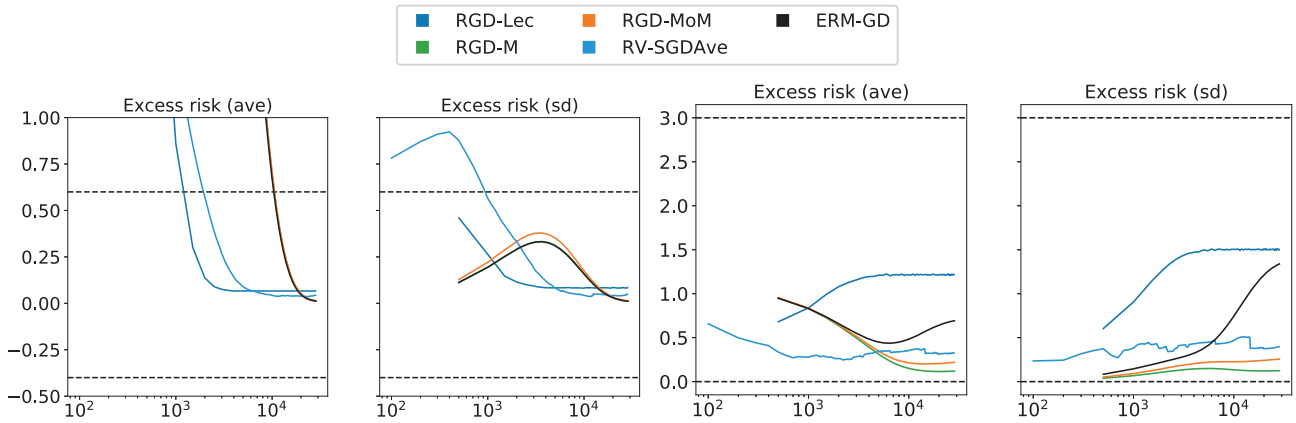

Figure 6: Comparison with robust GD methods. Left: Normal case. Right: log-Normal case.

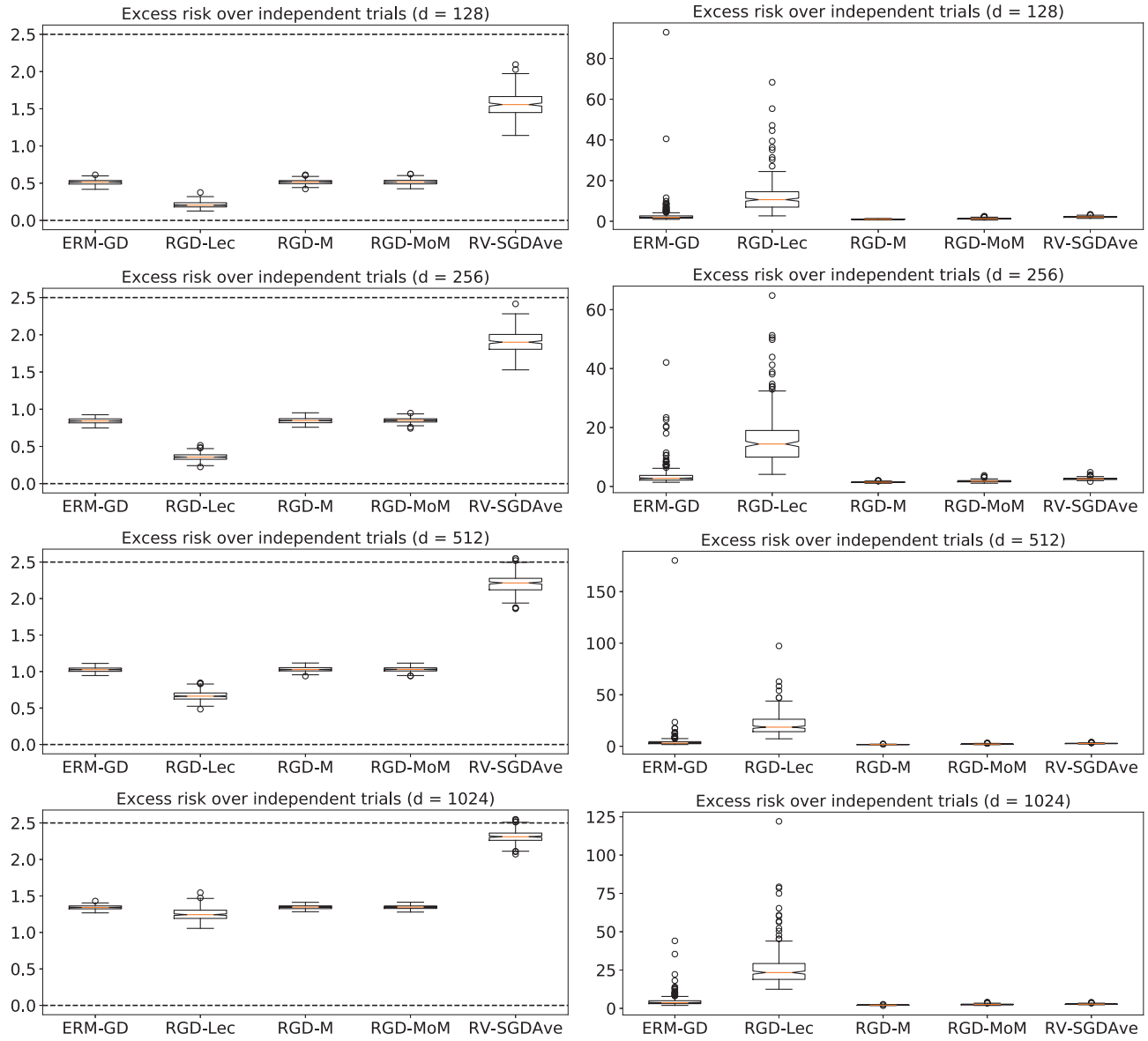

Figure 7: Excess risk for many-pass batch methods and single-pass `RV-SGDAve`. Dimension settings shown are $d \in \{128, 256, 512, 1024\}$. Left column: Normal noise (dashed horizontal rule is fixed to show small relative changes). Right column: log-Normal noise.

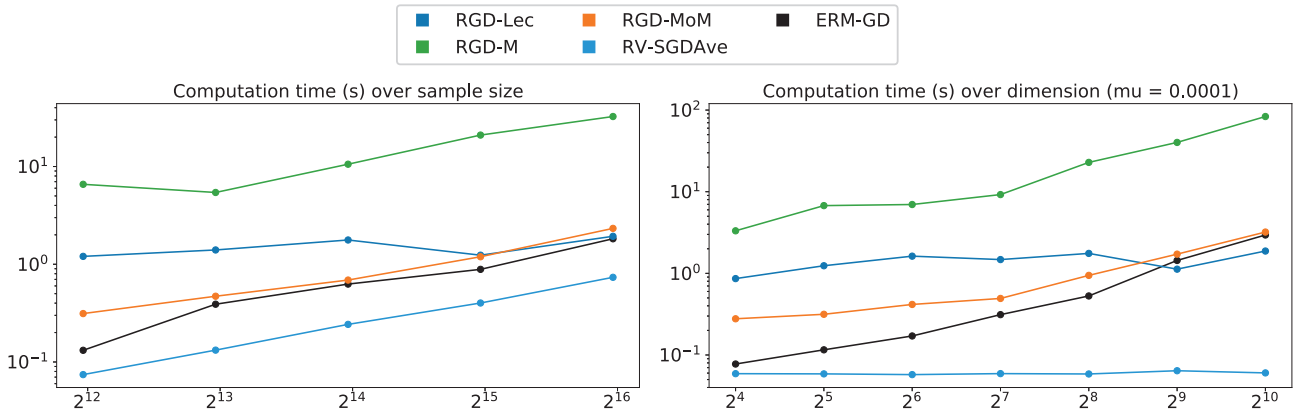

Figure 8: Median computation times (log scale, base 10) as a function of n and d (right; log scale, base 2). Left: time as a function of n (log scale, base 2), with n and d growing together. Right: time as a function of d (log scale, base 2), with n fixed.

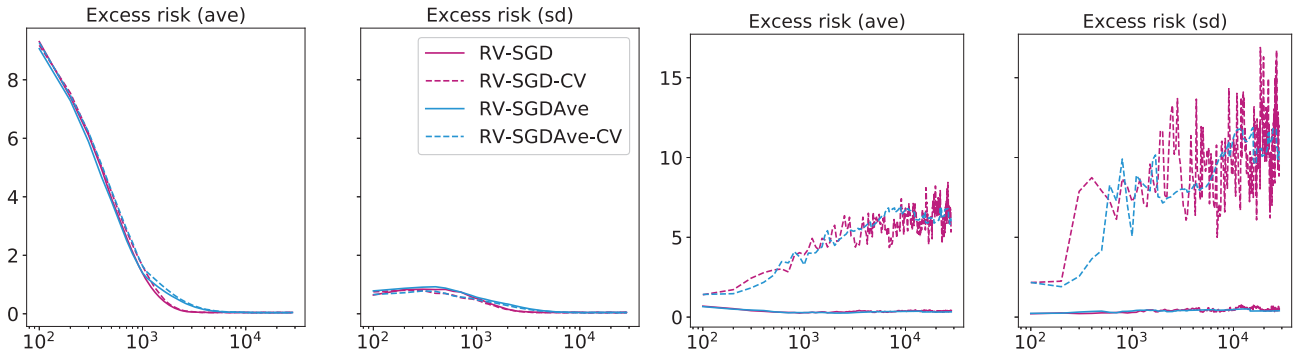

Figure 9: The negative impact of trying to modify Algorithm 1 to use a cross validation heuristic. Left: Normal noise. Right: log-Normal noise.