Scaling-Up Robust Gradient Descent Techniques

Matthew J. Holland
Osaka University
matthew-h@ar.sanken.osaka-u.ac.jp

Abstract

We study a scalable alternative to robust gradient descent (RGD) techniques that can be used when losses and/or gradients can be heavy-tailed, though this will be unknown to the learner. The core technique is simple: instead of trying to robustly aggregate gradients at each step, which is costly and leads to sub-optimal dimension dependence in risk bounds, we choose a candidate which does not diverge too far from the majority of cheap stochastic sub-processes run over partitioned data. This lets us retain the formal strength of RGD methods at a fraction of the cost.

Introduction

Obtaining "strong contracts" for the performance of machine learning algorithms is difficult.[1] Classical tasks in computer science, such as sorting integers or simple matrix operations, come with lucid worst-case guarantees. With enough resources, the job *can* be done correctly and completely. In machine learning, since we only have access to highly impoverished information regarding the phenomena or goal of interest, inevitably the learning task is uncertain, and any meaningful performance guarantee can only be stated with some degree of confidence, typically over the random draw of the data used for training. This uncertainty is reflected in the standard formulation of machine learning tasks as "risk minimization" problems (Vapnik 1982; Haussler 1992). Here we consider risk minimization over some set of candidates $\mathcal{W} \subseteq \mathbb{R}^d$, where the *risk* of w is defined as the expected loss to be incurred by w, namely

$$R_{\mathrm{P}}(w) := \mathbf{E}_{\mathrm{P}}\, L(w;Z) = \int_{\mathcal{Z}} L(w;z)\,\mathrm{P}(\mathrm{d}z), \qquad w \in \mathcal{W}.$$

Here we have a loss function $L : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}_+$, and random data $Z \sim \mathrm{P}$ takes values in a set $\mathcal{Z}$. At most, any 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

$$\mathbf{P}\left\{R_{\mathrm{P}}(\widehat{w}_n) - R^*_{\mathrm{P}} > \varepsilon\left(n, \delta, \mathrm{P}, \mathcal{W}\right)\right\} \leq \delta, \qquad \forall\, \mathrm{P} \in \mathcal{P}.$$

Flipping the inequalities around, this says that the algorithm generating $\widehat{w}_n$ enjoys ε-good performance with $(1-\delta)$-high confidence over the draw of the sample, where the error level depends on the sample size n, the desired confidence level δ, the underlying data distribution P, and any constraints encoded in $\mathcal{W}$, not to mention the nature of loss L. Ideally, we would like formal guarantees to align as closely as possible with performance observed in the real world by machine learning practitioners. With this in mind, the following properties are important to consider.

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*. If strong assumptions are made on the data distribution (i.e., $\mathcal{P}$ is a "small" class), then most of the data any practitioner runs into will fall out of scope. If the error bound grows too quickly with $1/\delta$ or shrinks too slowly with n, then either the guarantees are vacuous, or the procedure is truly sub-optimal. If the procedure outputting $\widehat{w}_n$ cannot be implemented, then we run into a gap between what we code, and what we study formally.

Our problem setting In this work, we consider the setup of *potentially heavy-tailed* data, specialized to the case of strongly convex loss functions; in related work (Holland 2021), we consider a completely different approach when we do not have strong convexity. More concretely, all the learner can know is that for some $m < \infty$,

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

[1]This notion was described lucidly in a keynote lecture by L. Bottou (Bottou 2015).

where typically $m = 2$. Thus, it is unknown whether the losses (or partial derivatives, etc.) are congenial in 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. We next review the related technical literature, and give an overview of our contributions.

Context and Contributions

With the three properties of transparency, strength, and scalability highlighted in the previous section in mind, for the next few paragraphs we look at the characteristics of several important families of learning algorithms.

ERM: can scale well, but lacks robustness Classical learning theory is primarily centered around *empirical risk minimization* (ERM) (Vapnik 1998; Anthony and Bartlett 1999), and studies the statistical properties that hold for *any* minimizer of the empirical risk, namely

$$\widehat{w}_n \in \operatorname*{arg\,min}_{w \in \mathcal{W}} \frac{1}{n} \sum_{i=1}^{n} L(w; Z_i). \tag{2}$$

Clearly, this leaves all algorithmic aspects of the problem totally abstract, and opens up the possibility for substantial gaps between the performance of "good" and "bad" ERM solutions, as studied by Feldman (2017). Furthermore, the empirical mean is sensitive to outliers, and formally speaking is sub-optimal in the sense that it cannot achieve sub-Gaussian error bounds under potentially heavy tails, while other practical procedures can; see Catoni (2012) and Devroye et al. (2016) for comprehensive studies. Roughly speaking, the empirical mean cannot guarantee better error bounds than those which scale as $\Omega(1/\sqrt{\delta n})$. In the context of machine learning, these statistical limitations provide an important implication about the *feedback* available to any learner which tries to directly minimize the empirical risk, effectively lower-bounding the statistical error (in contrast to the optimization error) incurred by any such procedure.

Robust risk minimizers: strong in theory, but lacks transparency To deal with the statistical weaknesses of ERM, it is natural to consider algorithms based on more "robust" feedback, i.e., minimizers of estimators of the risk which provide stronger guarantees than the empirical mean under potentially heavy tails. A seminal example of this is the work of Brownlees, Joly, and Lugosi (2015), who consider learning algorithms of the form

$$\widehat{w}_n \in \operatorname*{arg\,min}_{w \in \mathcal{W}} \widehat{R}(w), \text{ s.t. } \sum_{i=1}^{n} \psi\left(\frac{\widehat{R}(w) - L(w; Z_i)}{s}\right) = 0. \tag{3}$$

That is, they consider minimizers of an M-estimator of the risk, using influence function ψ of the type studied by Catoni (2012). 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 compared with 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. Observe that the new objective $\widehat{R}(\cdot)$ cannot be written in closed form, and even if $L(\cdot; Z)$ is convex, this $\widehat{R}(\cdot)$ need not preserve such convexity. Direct optimization is hard, but verifying improvement in the function value is easy, and some researchers have utilized a guess-and-check strategy to make the approach viable in practice (Holland and Ikeda 2017). However, these methods are inexact, and due to optimization error, strictly speaking the algorithm being run does not enjoy the full guarantees given by Brownlees, Joly, and Lugosi (2015) for the ideal case.

Robust gradient descent: transparent, but scales poorly To try and address the issue of transparency without sacrificing the strength of formal guarantees, 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), the naming being appropriate since their core updates all take the form

$$\widehat{w}_{t+1} = \widehat{w}_t - \alpha_t \, \widehat{G}_n(\widehat{w}_t), \tag{4}$$

and they are "robust" in the sense that the estimate $\widehat{G}_n(w) \approx \nabla R_{\mathrm{P}}(w)$ has deviations with near-optimal confidence intervals under potentially heavy-tailed data (i.e., both the loss and partial gradients are potentially heavy-tailed). The most common strategy replaces the empirical risk gradient with a high-dimensional "median of means" estimate (RGD-by-MoM) (Chen, Su, and Xu 2017b,a; Prasad et al. 2018; El-Mhamdi et al. 2019; Rajput et al. 2020). Other strategies involve dimension-wise truncation (Holland 2019) and M-estimation of the risk partial derivatives (RGD-M) (Holland and Ikeda 2019). Note that both approaches enjoy error bounds with optimal dependence on n and $1/\delta$ under potentially heavy-tailed data, with the significant merit that the computational procedures are transparent and easy to implement as-is. Unfortunately, instead of a simple one-dimensional robust mean estimate as in (3), 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, hampering their overall scalability.

Summarizing the above points, first of all, ERM and robust risk minimizers leave the potential for a severe gap between what is guaranteed on paper and what is done in practice. On the other hand, both formal guarantees and computational requirements for RGD methods do not scale well to high-dimensional learning tasks. For comparison, we show concrete error bounds for RGD procedures under strongly convex losses in Table 1. The key issues are clear: even when working with the Euclidean geometry, a quick glance at the proofs in the cited works on RGD shows that direct dependence on d in the error bounds is unavoidable. Furthermore, the extra computational overhead, scaling at least linearly in

Method	Error	Cost
DC-SGD (Algorithm 1)	$\mathcal{O}\left(\frac{\log(\delta^{-1})}{n}\right)$	$\mathcal{O}\left(\frac{dn}{\log(\delta^{-1})}\right) + \text{cost}\,(\texttt{Merge})$
RGD-by-MoM (Chen et al. 2017a)	$\mathcal{O}\left((1-c)^{2T}\right) + \mathcal{O}\left(\frac{k(d+\log(\delta^{-1}))}{n}\right)$	$\mathcal{O}\left(\frac{Tdn}{k}\right) + T\,\text{cost}(\texttt{GeoMed})$
RGD-M (Holland and Ikeda 2019)	$\mathcal{O}\left((1-c)^{2T}\right) + \mathcal{O}\left(\frac{d(\log(d\delta^{-1})+\log(n))}{n}\right)$	$\mathcal{O}\left(Tdn\right)$

Table 1: Here we compare performance guarantees for different learning algorithms. Error refers to $1-\delta$ confidence intervals for $R_{\mathrm{P}}(\cdot) - R_{\mathrm{P}}^{*}$, evaluated at the output of each algorithm after T iterations (with DC-SGD using $T = n$ by definition). All error bounds stated are under the assumptions of Theorem 1. Here GeoMed refers to the geometric median sub-routine (Minsker 2015; Vardi and Zhang 2000). Cost estimates assume the availability of k cores for parallel computations. See appendix for additional details.

d, must be incurred at *every* step in the iterative procedure, which severely hurts scalability.

Our contributions Considering the issues highlighted above, in particular with the scalability of modern RGD techniques in terms of loose guarantees and computational cost, here we investigate a different algorithmic approach of equal generality, with the goal of achieving as-good or better dependence on n, d, and $1/\delta$, under the same assumptions, and in provably less time for larger problems. The core technique uses distance-based rules to select among independent weak candidates, which are implemented using inexpensive stochastic gradient-based updates. Our main contributions:

- We analyze a flexible and robust archetype for learning under heavy tails (Algorithm 1), and obtain sharp high-probability error bounds (Theorem 1) that improve on the poor dimension dependence of existing RGD routines under strongly convex risks when both the losses and gradients can be heavy-tailed (see Table 1).
- The procedure outlined in Algorithm 1 is simple to implement and amenable to distributed computation, providing superior computational scalability over existing serial RGD procedures, without sacrificing the strength or transparency of theoretical guarantees.
- Empirically, we study the efficiency and robustness of the proposed algorithm and its key competitors in a tightly controlled simulated setting (section), verifying a substantial improvement in the cost-performance tradeoff, robustness to heavy-tailed data, and performance that scales well to higher dimensions.

Taken together, our results suggest a promising class of algorithms for convex risk minimization, which achieve an appealing balance between transparency, strength and scalability.

Theoretical Analysis

Preliminaries

First we establish some basic notation, and organize technical details in one place for ease of reference.

Notation 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 $\sigma_{\mathrm{P}}^2(w) := \mathrm{var}_{\mathrm{P}}\, L(w;Z) = \mathbf{E}_{\mathrm{P}}(L(w;Z) - R_{\mathrm{P}}(w))^2$ for each $w \in \mathcal{W}$. When we write $I\{\texttt{event}\}$, this refers to the indicator function which returns 1 when event is true, and 0 otherwise.

Technical conditions 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. 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, and μ-*strongly convex* if $\langle u-v, \nabla f(u) - \nabla f(v)\rangle \geq \mu\|u-v\|^2$ for all u, v (details given in appendix). Finally, to make formal statements technically simpler, we assume that $R_{\mathrm{P}}(\cdot)$ achieves its minimum on the interior of $\mathcal{W}$.

Overview of algorithm We study a general-purpose learning algorithm using a divide-and-conquer strategy, summarized in Algorithm 1. Following a partition of the data into k disjoint subsets, a sub-routine SGD is used to generate k candidates, and from these candidates, a final output is determined by Merge, a generic sub-routine whose desirable properties will be discussed shortly (see (7)). As for SGD, for concreteness we are considering traditional (projected) stochastic gradient descent. The core update of arbitrary point w given data $Z \sim \mathrm{P}$ is given by

$$\texttt{SGD}\,[w; Z, \alpha, \mathcal{W}] := \Pi_{\mathcal{W}}\left(w - \alpha\, G(w;Z)\right). \quad (5)$$

Algorithm 1 Robust divide and conquer archetype; $\texttt{DC-SGD}\,[\boldsymbol{Z}_n, \widehat{w}_0; k]$.

```
inputs: sample $\boldsymbol{Z}_n$, initial value $\widehat{w}_0 \in \mathcal{W}$, 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 $j \neq l$.    ▷ Disjoint partition.
$\widehat{w}^{(j)} = \texttt{SGD}\left[\widehat{w}_0; \boldsymbol{Z}_{\mathcal{I}_j}, \mathcal{W}\right]$, for each $j \in [k]$.    ▷ Obtain inexpensive candidates.
return: $\widehat{w}_{\texttt{DC}} = \texttt{Merge}\left[\{\widehat{w}^{(1)}, \ldots, \widehat{w}^{(k)}\}; \|\cdot\|_2\right]$.    ▷ Follow the majority.
```

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 $w \in \mathcal{W}$. That is, we assume access to an unbiased estimate of some sub-gradient of the true risk. For an arbitrary sequence $(Z_1, Z_2, \ldots, Z_t)$ of length $t \geq 1$, let $\texttt{SGD}[\widehat{w}_0; (Z_1, \ldots, Z_t), \mathcal{W}] := \texttt{SGD}[\widehat{w}_{t-1}; Z_t, \alpha_{t-1}, \mathcal{W}]$. Note that using (5), the right-hand side is defined recursively, and bottoms out at $t = 0$, using pre-fixed initial value $\widehat{w}_0$. Note that we suppress the step sizes $(\alpha_0, \ldots, \alpha_{t-1})$ from this notation for readability. For any arbitrary sub-index $\mathcal{I} \subseteq [n]$, sequence $\texttt{SGD}[\widehat{w}_0; \boldsymbol{Z}_{\mathcal{I}}, \mathcal{W}]$ is defined analogously; since the Z_i are iid, the sequence order does not matter.

Error Bounds Under Heavy-Tailed Losses and Gradients

We start by stating the main performance guarantee for Algorithm 1, which holds for potentially heavy-tailed losses/gradients, and then sketch out the core principles behind the proof.

Theorem 1. *Let the risk be μ-strongly convex, and the loss be λ_1-smooth. Run Algorithm 1 with a sample size at least $n \geq \max\{k, M^*\}$, where $k = \lceil 8 \log(\delta^{-1}) \rceil$, and*

$$M^* := \frac{4\lambda_1}{\mu} \left(\max\left\{ \frac{\lambda_1 \mu \|\widehat{w}_0 - w^*\|^2}{\mathbf{E}_{\mathrm{P}}\, \|G(w^*; Z)\|^2}, 1 \right\} - 1 \right).$$

For the initial update set $\alpha_0 = 1/(2\lambda_1)$, and subsequent step sizes $\alpha_t = a/(\mu n + b)$ for $t > 0$, with $b = 2a\lambda_1$, and $a > 0$ set such that $\alpha_t \leq \alpha_0$ for all t. Then, with probability no less than $1 - \delta$, we have

$$R_{\mathrm{P}}(\widehat{w}_{\texttt{DC}}) - R_{\mathrm{P}}^* \leq \mathbf{E}_{\mathrm{P}}\, \|G(w^*; Z)\|^2 \left(\frac{a\lambda_1}{\mu} \right)^2 \frac{2c \log(\delta^{-1})}{n - M_\delta}$$

where $M_\delta \leq 16 \log(\delta^{-1})(M^* - b)$, and c depends only on the choice of $\texttt{Merge}$.

Proving such a theorem is straightforward using the quadratic growth property of strongly convex functions in conjunction with λ_1-smoothness. When the risk is μ-strongly convex, we have the critical property that points which are ε-far away from the minimum w^* must be $(\varepsilon^2 \mu/2)$-bad in terms of excess risk. As such, simple distance-based robust aggregation metrics can be used to efficiently boost the confidence. To start, we need a few basic facts which will be used to characterize a valid $\texttt{Merge}$ operation.[2] Given k points $u_1, \ldots, u_k \in \mathbb{R}^d$, the basic requirement here is that we want the output of $\texttt{Merge}$ to be close to the majority of these points, in the appropriate norm. To make this concrete, define

$$\Delta(u; \gamma, \{u_1, \ldots, u_k\}, \|\cdot\|) := \inf\left\{ r \geq 0 : |\{j : \|u_j - u\| \leq r\}| > k\left(\frac{1}{2} + \gamma\right) \right\}. \quad (6)$$

When the other parameters are obvious from the context, we shall write simply $\Delta(u; \gamma) = \Delta(u; \gamma, \{u_1, \ldots, u_k\}, \|\cdot\|)$. In words, $\Delta(u; \gamma)$ is the radius of the smallest ball centered at u which contains a γ-majority of the points $u_1, \ldots, u_k$. Using this quantity, our requirement on $\texttt{Merge}$ is that for any $0 \leq \gamma < 1/2$ and $u \in \mathbb{R}^d$, we have

$$\|\widehat{u} - u\| \leq c_\gamma \Delta(u; \gamma, \{u_1, \ldots, u_k\}), \quad \text{where } \widehat{u} = \texttt{Merge}\left[\{u_1, \ldots, u_k\}; \|\cdot\|\right]. \quad (7)$$

Here c_γ is a factor that is independent of the choice of u or the points $u_1, \ldots, u_k$ given, which depends only on the choice of γ.

Next, considering the partitioning scheme of Algorithm 1, the ideal case is of course where, given some desired performance level $R_{\mathrm{P}}(\widehat{w}^{(j)}) - R_{\mathrm{P}}^* \leq \varepsilon$, the $\texttt{SGD}$ sub-routine returns an ε-good candidate for all $j \in [k]$ subsets. In practice, we will not always be so lucky, but the following lemma shows that with enough candidates, most of them will be ε-good with high confidence.

Lemma 2. *Let $(S, \|\cdot\|)$ be any normed linear space. Let $X_1, \ldots, X_k$ be iid random entities taking values in S, and fix $x^* \in S$. For $\varepsilon > 0$, write $a_i(\varepsilon) := I\{\|X_i - x^*\| \leq \varepsilon\}$, $\delta_\varepsilon := 1 - \mathbf{E}\, a(\varepsilon)$. For any $0 \leq \gamma < (1/2 - \delta_\varepsilon)$, it follows that*

$$\mathbf{P}\left\{ \sum_{i=1}^{k} a_i(\varepsilon) > k\left(\frac{1}{2} + \gamma\right) \right\} \geq 1 - \mathrm{e}^{-2k\left(\gamma + \delta_\varepsilon - \frac{1}{2}\right)^2}.$$

Applying Lemma 2 using the event $a_j(\varepsilon) = I\{R_{\mathrm{P}}(\widehat{w}^{(j)}) - R_{\mathrm{P}}^* \leq \varepsilon\}$, we see that when k scales with $\log(\delta^{-1})$, we can guarantee that there is a $1 - \delta$ probability good event in which at least a γ-majority of the candidates are ε-good. On this good event, via strong convexity it follows that a γ-majority of the candidates are $\sqrt{2\varepsilon/\mu}$-close to w^*, which means $\Delta(w^*; \gamma, \{\widehat{w}^{(1)}, \ldots, \widehat{w}^{(k)}\}) \leq \sqrt{2\varepsilon/\mu}$. Leveraging the requirement (7) on $\texttt{Merge}$, one obtains the following general-purpose boosting procedure.

[2]Procedures with this property are called "robust distance approximation" by Hsu and Sabato (2016).

Lemma 3 (Boosting the confidence, under strong convexity)**.** *Assume the risk is μ-strongly convex and λ_1-smooth, and that we have a learning algorithm $\widehat{w}_{\text{OLD}}$ which for $n \geq 1$ and $\delta_0 \in (0,1)$ achieves*

$$\mathbf{P}\left\{R_{\mathrm{P}}(\widehat{w}_{\text{OLD}}) - R_{\mathrm{P}}^{*} > \frac{\varepsilon_{\mathrm{P}}(n)}{\delta_0}\right\} \leq \delta_0.$$

For desired confidence level δ, split $\boldsymbol{Z}_n$ into $k = \lceil 8\log(\delta^{-1})/(1-\gamma)^2 \rceil$ disjoint subsets, let $\widehat{w}_{\text{OLD}}^{(1)}, \ldots, \widehat{w}_{\text{OLD}}^{(k)}$ denote the outputs of $\widehat{w}_{\text{OLD}}$ run on these subsets, and construct $\widehat{w}_{\text{NEW}}$ as

$$\widehat{w}_{\text{NEW}} = \texttt{Merge}\left[\{\widehat{w}_{\text{OLD}}^{(1)}, \ldots, \widehat{w}_{\text{OLD}}^{(k)}\}; \|\cdot\|\right].$$

As long as `Merge` *satisfies (7), then for any $0 \leq \gamma < 1/4$ and $n \geq k$, we have that*

$$R_{\mathrm{P}}(\widehat{w}_{\text{NEW}}) - R_{\mathrm{P}}^{*} \leq \frac{4c_{\gamma}^{2}\lambda_1}{\mu}\varepsilon_{\mathrm{P}}\left(\frac{(1-\gamma)^2 n}{8\log(\delta^{-1})}\right)$$

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

With these basic results in place, we can readily prove Theorem 1.

Proof of Theorem 1. Using the assumptions provided in the hypothesis, we can obviously leverage Lemma 3. The correspondence between this lemma and Algorithm 1 is $\widehat{w}_{\text{OLD}}^{(j)} \leftrightarrow \widehat{w}^{(j)}$ and $\widehat{w}_{\text{NEW}} \leftrightarrow \widehat{w}_{\text{DC}}$. The remaining task is to fill in $\varepsilon_{\mathrm{P}}(\cdot)$ for the final iterate of standard SGD, using the prescribed step sizes. It is well-known that for *averaged* SGD, one does not need to require that the losses be Lipschitz. On the other hand, for last-iterate SGD, it was only quite recently that Nguyen et al. (2018), in a nice argument building upon Bottou, Curtis, and Nocedal (2016), showed that the Lipschitz condition is not required if we have λ_1-smooth losses. For our purposes, this implies that

$$\varepsilon_{\mathrm{P}}(m) \leq \frac{\mathbf{E}_{\mathrm{P}}\|G(w^*;Z)\|^2}{m - M^* + b}\left(\frac{2a^2\lambda_1}{\mu}\right)$$

for any choice of $m \geq M^*$. A detailed statement of the more general property used is given in Theorem 7 in the appendix. Applying this to Lemma 3 for Algorithm 1, we shall have $m = (1-\gamma)^2 n/8\log(\delta^{-1})$. Multiplying these factors out, we end up with $n - 8\log(\delta^{-1})(M^*-b)/(1-\gamma)^2$ in the denominator, for arbitrary choice of $0 \leq \gamma < 1/4$. To cover all choices of `Merge` and thus γ, we simply use the rough upper bound $8\log(\delta^{-1})(M^*-b)/(1-\gamma)^2 \leq 16\log(\delta^{-1})(M^*-b)$ in the stated result. $\square$

Concrete implementations of `Merge` There are many natural choices for implementing `Merge`. For example, the geometric median (minimizing the sum of absolute deviations) can be easily implemented (Vardi and Zhang 2000; Cohen et al. 2016), and enjoys a factor of $c_{\gamma} \leq (1 + 1/(2\gamma))$ (Minsker 2015). A simple smallest-ball procedure, which takes the point that contains a γ-majority in the smallest-radius ball just requires doing pairwise distance calculations, and satisfies $c_{\gamma} \leq 3$ (Hsu and Sabato 2016). Using a coordinate-wise median approach is the easiest to code, but introduces dimension dependence, as $c_{\gamma} \leq \sqrt{d}(1 + 1/(2\gamma))$. Plugging these upper bounds into the proof of Theorem 1, by basic arithmetic, the `Merge`-dependent constant c can be bounded as $c \leq 1536$, $c \leq 288$, and $c \leq 1536d$ respectively.

Additional related literature The excess risk bounds given by Theorem 1 give us an example of the guarantees that are possible under potentially heavy-tailed data, for arguably the simplest divide-and-conquer strategy one could conceive of. Here we remark that the core idea of using robust aggregation methods to boost the confidence of independent candidates under potentially heavy-tailed data can be seen in various special cases throughout the literature. For example, influential work from Minsker (2015, Sec. 4) applies the geometric median to robustify both PCA and high-dimensional linear regression procedures, under potentially heavy-tailed observations. Hsu and Sabato (2016, Sec. 4.2) look at merging ERM solutions when the *empirical* risk is strongly convex, using a smallest-ball strategy. In contrast, we do not require the losses to be strongly convex, and our computational procedure is explicit, yielding bounds which incorporate error of both a statistical and computational nature, unlike ERM-type guarantees.

Empirical Analysis

In this section, we use controlled simulations to investigate how the differences in formal performance guarantees discussed in the previous section work out in practice.

Experimental setup We essentially follow the "noisy convex minimization" tests used in the literature to test the robustness of RGD methods (Holland and Ikeda 2019). Complete details of the experimental setup are provided in the supplementary materials.[3] 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 control the μ-strong convexity of the risk and ensure the gradients $\nabla L(w;Z) = -(\langle w^* - w, X\rangle + E)X$ are Lipschitz continuous, satisfying the two key conditions of Theorem 1.

Regarding the methods being compared, as classical baselines, empirical risk minimization using batch GD (denoted `ERM-GD`) and stochastic GD (denoted `SGD`) are used. We also implement standard RGD methods as more modern benchmarks: RGD-by-MoM (Chen, Su, and Xu 2017a; Prasad et al. 2018) (denoted here as `RGD-MoM`), RGD-M (Holland and Ikeda 2019) (`RGD-M`), and median-of-means minimization by gradient descent of Lecué, Lerasle, and Mathieu (2018) (`RGD-Lec`). Against these methods, we compare Algorithm 1 (`DC-SGD`), with `Merge` computed us-

[3]Repository: https://github.com/feedbackward/sgd-roboost

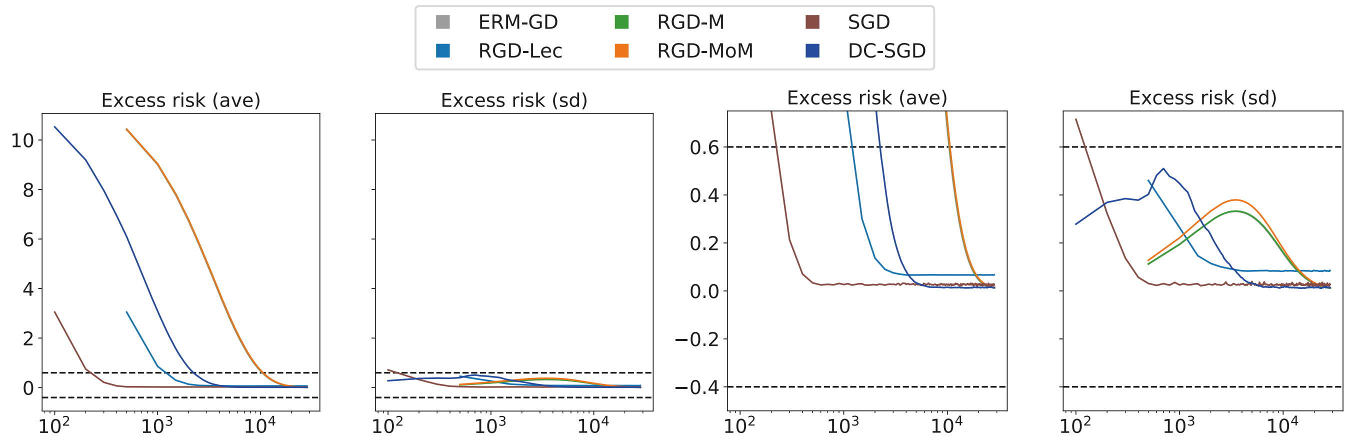


Figure 1: Excess risk statistics as a function of cost in gradient computations (log scale, base 10). The two right-most plots zoom in on the region between the dashed lines in the two left-most plots.

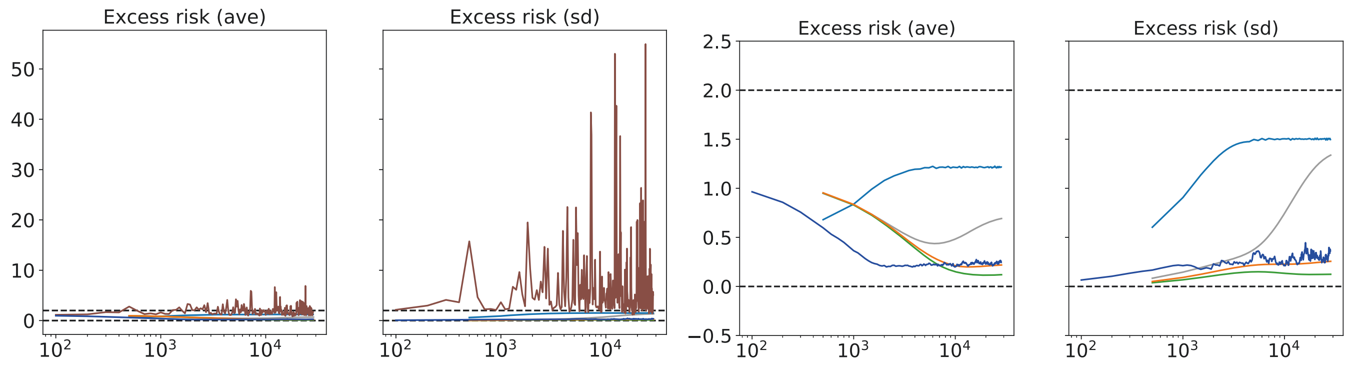


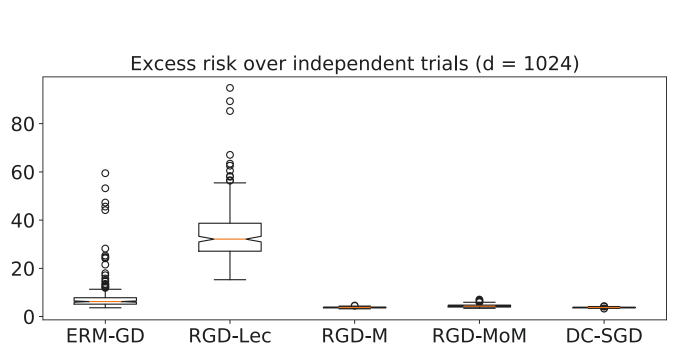
Figure 2: Analogous results to Figure 1, for the case of log-Normal noise. Note for the zoomed-in plots on the right, we have removed the volatile SGD trajectory for visibility.

ing the geometric median (Vardi and Zhang 2000). All detailed settings are in the supplementary materials.

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. 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. For convenience, here we list the key contents of our empirical analysis (extra figures in supplementary materials):

1. Error trajectories in low dimensions (fixed n and d, many iterations).
2. Statistical error in high dimensions (d grows, n fixed).
3. Actual computation times (d grows, n fixed/grows).
4. Impact of initialization on error trajectories ($\|\widehat{w}_0 - w^{*}\|$ grows).
5. Impact of noise level on error trajectories (signal/noise ratio gets worse).

Discussion of results Plots of representative empirical test results are in Figures 1–4. We start with low-dimensional tests to examine the nature of the error trajectory of Algorithm 1 (`DC-SGD`), when run for multiple passes over the data. With no fine-tuning of algorithm parameters, by simply doing a proper aggregation of noisy SGD sub-processes, it is clearly possible to achieve performance comparable to well-tuned RGD methods, and do so using far less computational resources, both on average and in terms of between-trial variance (Figures 1–2). Clearly, even when the underlying sub-processes used by Algorithm 1 are very noisy, only a few passes over the data are necessary to match the best-performing RGD methods, both on average and in terms



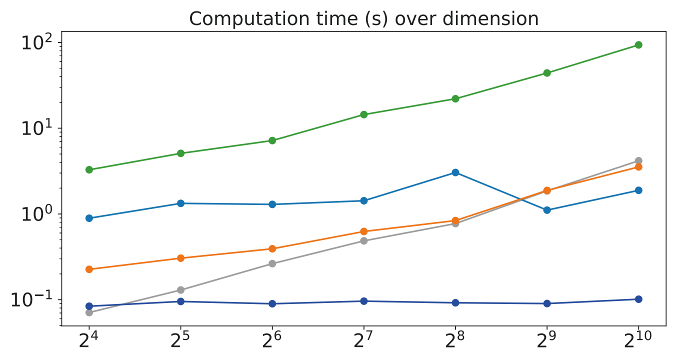


Figure 3: Left: box-plot of final excess risk for batch methods (many-pass) versus DC−SGD (two-pass), with $d = 1024$ under log-Normal noise. Right: median computation times as d increases.

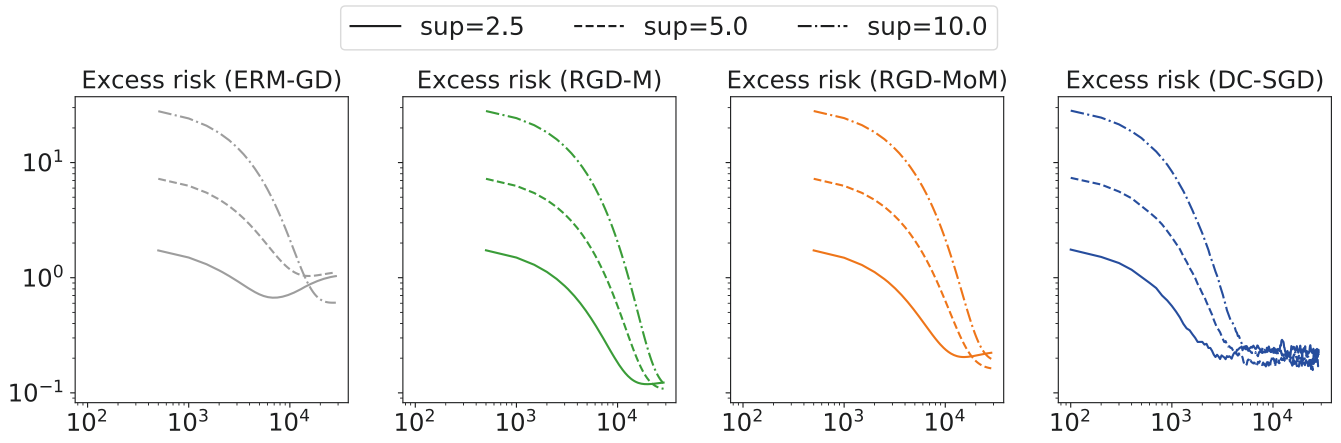


Figure 4: Excess risk trajectories (averaged over trials) with different initialization error ranges. Here sup refers to the error range, where $\widehat{w}_{0,j} = w_j^* + \text{Uniform}[-\texttt{sup}, +\texttt{sup}]$ for each $j \in [d]$.

of between-trial variance. Furthermore, without any algorithm adjustments, this robustness holds over changes to the signal/noise ratio and initialization error (Figure 4). While sample-splitting can cause our procedure to take a small hit in statistical error for very large n, it makes up for this in scalability as d grows. Even with just two passes over the data, at the order of thousands of parameters, the proposed procedure is able to achieve comparable performance under well-behaved gradients, and superior performance under heavy-tailed gradients, without prior knowledge or retuning, and at a fraction of the cost (Figure 3).

Future Directions

This paper presents evidence, both formal and empirical, that a general-purpose learning algorithm following the archetype drawn out in Algorithm 1 should be able to improve significantly on the scalability of modern robust gradient descent methods under potentially heavy-tailed losses and gradients, without sacrificing formal guarantees. Extending the theory to other algorithms besides vanilla SGD is a straightforward exercise; less straightforward is when we start considering a stage-wise strategy, when partition size k can change from stage to stage. Extending results to allow for multiple passes is also of natural interest; the work of Lin and Rosasco (2017) does this for the squared error, but without heavy tails. We only covered the ℓ_2 norm case here, but extensions to cover other geometries (via stochastic mirror descent for example) are also of interest. The RGD methods cited in this work are chiefly designed for convex optimization problems; they sacrifice exploration in favor of exploitation, and it will be interesting to investigate how approaches like Algorithm 1 fare in non-convex settings due to their higher tendency to "explore."

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

Anthony, M.; and Bartlett, P. L. 1999. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

Bottou, L. 2015. Two Big Challenges in Machine Learning.

URL https://leon.bottou.org/talks/2challenges. Keynote at ICML 2015.

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2016. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838* .

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

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. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*. ACM.

Chen, Y.; Su, L.; and Xu, J. 2017b. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *arXiv preprint arXiv:1705.05491v2* .

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*, 9–21.

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

El-Mhamdi, E.-M.; Guerraoui, R.; Guirguis, A.; and Rouault, S. 2019. SGD: Decentralized Byzantine Resilience. *arXiv preprint arXiv:1905.03853* .

Feldman, V. 2017. Generalization of ERM in Stochastic Convex Optimization: The Dimension Strikes Back. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, 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*, 703–711.

Holland, M. J. 2021. Robustness and scalability under heavy tails, without strong convexity. In *24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*, volume 130 of *Proceedings of Machine Learning Research*.

Holland, M. J.; and Ikeda, K. 2017. Robust regression using biased objectives. *Machine Learning* 106(9): 1643–1679. doi:10.1007/s10994-017-5653-5.

Holland, M. J.; and Ikeda, K. 2019. 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*.

Hsu, D.; and Sabato, S. 2016. Loss Minimization and Parameter Estimation with Heavy Tails. *Journal of Machine Learning Research* 17(18): 1–40.

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

Lin, J.; and Rosasco, L. 2017. Optimal Learning for Multi-pass Stochastic Gradient Methods. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, 4556–4564.

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

Nguyen, L. M.; Nguyen, P. H.; van Dijk, M.; Richtárik, P.; Scheinberg, K.; and Takáč, M. 2018. SGD and Hogwild! convergence without the bounded gradients assumption. *arXiv preprint arXiv:1802.03801v2* .

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

Rajput, S.; Wang, H.; Charles, Z.; and Papailiopoulos, D. 2020. DETOX: A Redundancy-based Framework for Faster and More Robust Gradient Aggregation. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 10320–10330.

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

Vapnik, V. N. 1998. *Statistical Learning Theory*. Wiley.

Vardi, Y.; and Zhang, C.-H. 2000. The multivariate L_1-median and associated data depth. *Proceedings of the National Academy of Sciences* 97(4): 1423–1426.