Robust Training in High Dimensions via Block Coordinate Geometric Median Descent

Anish Acharya, Abolfazl Hashemi
Prateek Jain, Sujay Sanghavi, Inderjit S. Dhillon & Ufuk Topcu

TL;DR

- While mini-batch SGD is the de-facto method to solve problems with **finite sum structure**; even a single grossly corrupted sample can lead SGD to an arbitrarily poor solution.
- The vulnerability of SGD is often attributed to the gradient aggregation using MEAN($\cdot$) which has 0 breakdown point. This motivates replacing the gradient aggregation with robust estimators like geometric median (GM), which *achieves the* ***optimal breakdown point*** *of* $\frac{1}{2}$.
- Despite the strong robustness guarantees of GM, all the known methods to compute an approximate GM solution, scale poorly with the dimension of the problem (d) making it prohibitive in high dimensional settings.
- We propose BGMD (Algorithm 1), a method for robust optimization in high dimensions, which is significantly more efficient than the standard GM-SGD method but is still able to maintain the optimal breakdown point.

Definition 1 (Gross Corruption Model). *Given $0 \le \psi < \frac{1}{2}$ and a distribution family $\mathcal{D}$ on $\mathbb{R}^d$ the adversary operates as follows: n samples are drawn from $D \in \mathcal{D}$. The adversary is allowed to* ***inspect*** *all the samples and replace up to ψn samples with arbitrary points. This implies that $\alpha := |\mathbb{B}|/|\mathbb{G}| < 1$, where $\mathbb{B}$ and $\mathbb{G}$ are the sets of corrupt and good samples.*

Definition 2 (Breakdown Point). *Breakdown point η of an estimator is the smallest fraction of contamination that must be introduced to cause an estimator to break e.g. an estimator has* **optimal breakdown point** *1/2 if robust $\forall$ $\alpha < 1$ gross corruption.*

Definition 3 (Geometric Median).

$$\mathbf{x}_* = \mathrm{GM}(\{\mathbf{x}_i\}) = \arg\min_{\mathbf{y}\in\mathbb{X}} \left[g(\mathbf{x}) := \sum_{i=1}^{n} \|\mathbf{y} - \mathbf{x}_i\| \right] \tag{1}$$

We call a point $\mathbf{x} \in \mathbb{R}^d$ an **ϵ-accurate geometric median** if $g(\mathbf{x}) \le (1+\epsilon) g(\mathbf{x}_*)$.

(Overview of BGmD). At a high level, at each iteration BGMD selects a block of $0 < k \le d$ *important* coordinates of the stochastic gradients using Algorithm 2. The remaining $(d-k)$ dimensions are discarded and gradient aggregation happens only along these selected k directions. While descending along only a small subset of k coordinates at each iteration significantly improves the per iteration computational cost (Lemma 2), a smaller value of k would also imply larger gradient information loss a smaller ξ (Lemma 1). Te mitigate this we propose a memory mechanism: Throughout training, we keep track of the residual error $\|\mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)\|$ incurred due to ignoring $(d-k)$ dimensions via $\hat{\mathbf{m}}_t \in \mathbb{R}^d$ that is appropriately added back in the subsequent iterations.

Algorithm 1 Block GM Descent (BGMD)

Initialize: estimate: $\mathbf{x}_0 \in \mathbb{R}^d$, step-size: γ, memory: $\hat{\mathbf{m}}_0 = \mathbf{0}$, Block Coordinate Selection operator: $\mathcal{C}_k(\cdot)$, Geometric Median operator: $\mathrm{GM}(\cdot)$
epochs t = 0, . . . , until convergence select samples $\mathcal{D}_t = \{i_1, \dots, i_b\}$
obtain: $\mathbf{g}_t^{(i)} := \nabla f_i(\mathbf{x}_t)$, $\forall i \in \mathcal{D}_t$ (back-propagation)
Let $\mathbf{G}_t \in \mathbb{R}^{b\times d}$ s.t. each row $\mathbf{G}_t[i,:] = \mathbf{g}_t^{(i)}$
$\mathbf{G}_t[i,:] \leftarrow \gamma \mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t \; \forall i \in [b]$ (add memory)
$\mathbf{\Delta}_t := \mathcal{C}_k(\mathbf{G}_t) \in \mathbb{R}^{b\times k}$ (subset k dim via Algo. 2)
$\mathbf{M}_{t+1} = \mathbf{G}_t - \mathbf{\Delta}_t$ (compute residuals)
$\hat{\mathbf{m}}_{t+1} = \frac{1}{b}\sum_{0\le i\le b} \mathbf{M}_{t+1}[i,:]$ (update memory)
$\tilde{\mathbf{g}}_t := \mathrm{GM}(\mathbf{\Delta}_t)$ (robust aggregation in $\mathbb{R}^k$)
$\mathbf{x}_{t+1} := \mathbf{x}_t - \tilde{\mathbf{g}}_t$ (parameter update)

Algorithm 2 Block Coordinate Selection Strategy

Input: $\mathbf{G}_t \in \mathbb{R}^{n\times d}$, k
coordinates j = 0, . . . , d-1 $s_j \leftarrow \|\mathbf{G}_t[:,j]\|^2$ (norm along each dimension) Sample set $\mathbb{I}_k$ of k dimensions with probabilities proportional to s_j
$\mathcal{C}_k(\mathbf{G}_t)[i, j \in \mathbb{I}_k] = \mathbf{G}_t[i,j]$, $\mathcal{C}_k(\mathbf{G}_t)[i, j \notin \mathbb{I}_k] = 0$
Return: $\mathcal{C}_k(\mathbf{G}_t)$

Convergence Guarantees

Lemma 1 (Contraction Mapping). Algorithm 2 yields a contraction approximation $\mathbb{E}\left[\|\mathcal{C}_k(\mathbf{G}_t) - \mathbf{G}_t\|^2 | \mathbf{G}_t\right] \le (1-\xi)\|\mathbf{G}_t\|^2$, $\frac{k}{d} \le \xi \le 1$.

Assumption 1 (Unbiased Oracle - Bounded Variance).

$$\mathbb{E}_{z\sim\mathcal{D}_i}[\mathbf{g}_i(\mathbf{x}, z)] = \nabla f_i(\mathbf{x}) \tag{2}$$

$$\mathbb{E}_{z\sim\mathcal{D}_i}\|\nabla F_i(\mathbf{x}, z)\|^2 \le \sigma^2 \tag{3}$$

Assumption 2 (Smoothness).

$$f_i(\mathbf{x}) \le f_i(\mathbf{y}) + \langle \mathbf{x} - \mathbf{y}, \nabla f_i(\mathbf{y})\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2 \tag{4}$$

Assumption 3 (Polyak-Lojasiewicz Condition).

$$\|\nabla f(\mathbf{x})\|^2 \ge 2\mu(f(\mathbf{x}) - f(\mathbf{x}^*)),\ \mu > 0 \tag{5}$$

Theorem 1 (Smooth Non-convex). *Suppose Assumption 1-2 hold. Run Algorithm 1 with compression factor ξ (Lemma 1), learning rate $\gamma_t = 1/2L$ and $\epsilon-$approximate $\mathrm{GM}(\cdot)$ in presence of $\alpha-$corruption (Definition 1) for T iterations, then for any $\tau \in [T]$ sampled uniformly at random:*

$$\mathbb{E}\|\nabla f(\mathbf{x}_\tau)\|^2 = \mathcal{O}\left(\frac{LR_0}{T} + \frac{\sigma^2\xi^{-2}}{(1-\alpha)^2} + \frac{L^2\epsilon^2}{|\mathbb{G}|^2(1-\alpha)^2}\right)$$

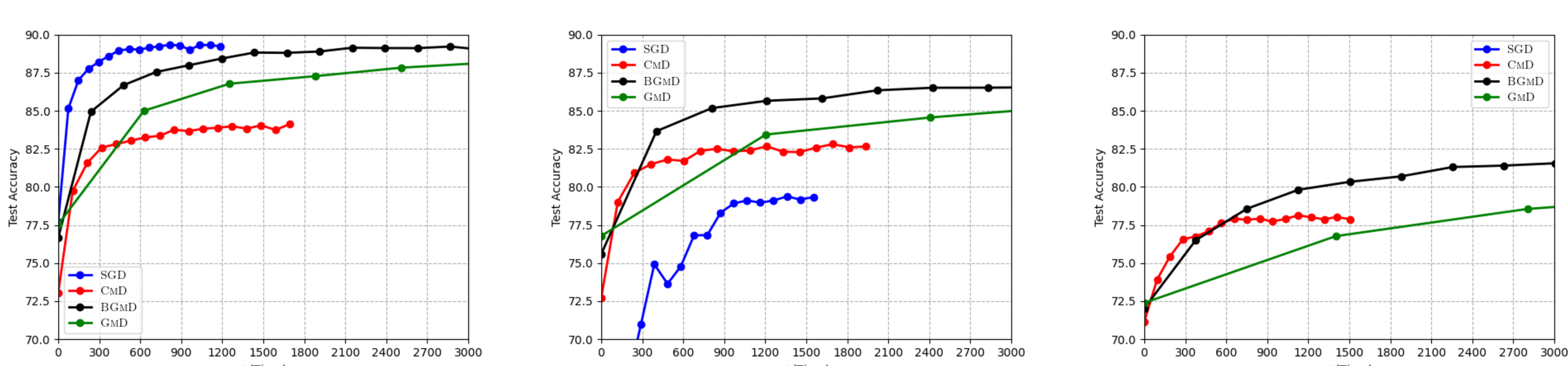
Theorem 2 (Non-convex under PLC). *Suppose Assumption 1-3 hold. Then, after T iterations BGMD with compression factor ξ, learning rate $\gamma_t = 1/4L$ and $\epsilon-$approximate $\mathrm{GM}(\cdot)$ oracle in presence of $\alpha-$corruption satisfies:*

$$\mathbb{E}\|\hat{\mathbf{x}}_T - \mathbf{x}^*\|^2 = \mathcal{O}\left(\frac{LR_0}{\mu^2}\left[1 - \frac{\mu}{8L}\right]^T + \frac{\sigma^2\xi^{-2}}{\mu^2(1-\alpha)^2} + \frac{L^2\epsilon^2}{\mu^2|\mathbb{G}|^2(1-\alpha)^2}\right)$$

where, $\hat{\mathbf{x}}_T := \frac{1}{W_T}\sum_{t=0}^{T-1} w_t \mathbf{x}_t$, $W_T := \sum_{t=0}^{T-1} w_t$ *with weights* $w_t := (1 - \frac{\mu}{8L})^{-(t+1)}$.

Empirical Evidence

Lemma 1 (Linear Speedup). For, $\beta \le \mathcal{O}(1/F - b\epsilon^2)$, given an ϵ- approximate GM oracle, Algorithm 1 achieves a factor F speedup over GM-SGD.



(a) No corruption (b) 20 % corruption (c) 40 % corruption

Robustness to Feature Corruption (Accuracy over clock time)

We observe with extensive experiments under different sources and models of corruption - Feature Corruption ; Gradient Corruption; Label Corruption; GM based methods are indeed superior while standard SGD or CMD can be significantly inaccurate. By judiciously choosing k, BGMD can be more efficient than GMD, often resulting in more than 3x speedup. Despite using small $\beta \le 0.15$ in all our experiments, it retains high generalization performance emphasizing the role of memory mechanism.