Block Coordinate Geometric Median Descent*

Anish Acharya

University of Texas at Austin

Facebook Ads ML Modeling
November 19, 2021

* Robust SGD in High Dimensions via Block Coordinate Geometric Median Descent.
Anish Acharya, Abolfazl Hashemi, Sujay Sanghavi, Inderjit Dhillon, Prateek Jain, Ufuk Topcu

TL;DR

- Geometric Median (GM) is a well studied rotation and translation invariant robust estimator with optimal **breakdown point** of 1/2 even under gross corruption.

- However high computational cost makes it infeasible for robust optimization in high dimensional settings.

- We show that by applying GM to only a judiciously chosen block of coordinates at a time and using a memory mechanism, one can retain the breakdown point of ½ while attaining the same non-asymptotic convergence rate as SGD with GM. We call the resulting algorithm BGMD.

- Empirically, BGMD can be up to 3x more efficient to train than GM-SGD while still ensuring similar test accuracy and maintaining same level of robustness.

- In clean setting, BGMD reaches similar accuracy as SGD while constrained to compute budget (see Fig. 2, 3, 4) indicating BGMD is a practical robust optimization approach in large scale settings.

Large Scale Optimization

- Training Deep Neural Network requires optimizing over highly over-parameterized , non-convex loss landscape.

- ERM Formulation - Finite Sum Structure

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left[\bar{f}(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) \right]$$

- Each function corresponds to loss over one sample

Stochastic Gradient Descent (SGD)

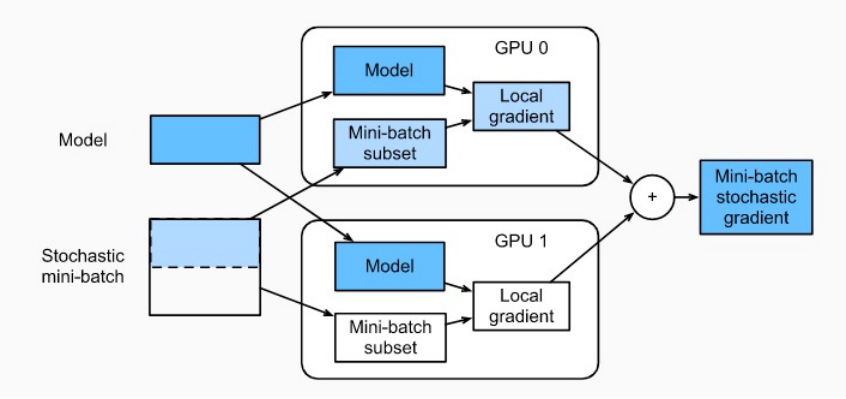
- Mini-batch SGD is the de-facto method for optimizing such functions

Initialize: estimate: $\mathbf{x}_0 \in \mathbb{R}^d$, step-size: γ
for *iterations* $t = 0, \ldots,$ *until convergence* **do**
  select samples $\mathcal{D}_t = \{i_1, \ldots, i_b\}$
  obtain: $\mathbf{g}_t^{(i)} := \nabla f_i(\mathbf{x}_t),\ \forall i \in \mathcal{D}_t$ (back-propagation)
  $\tilde{\mathbf{g}}_t := \frac{1}{b} \sum_{i=1}^{b} \mathbf{g}_t^{(i)}$ (gradient aggregation)
  $\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}_t$ (parameter update)
end

Distributed SGD

- Training happens over multiple compute nodes (e.g. GPU) in parallel
- Distributed Data Parallel

Initialize: estimate: $\mathbf{x}_0 \in \mathbb{R}^d$, step-size: γ
for *iterations* $t = 0, \ldots,$ *until convergence* **do**
  select samples $\mathcal{D}_t = \{i_1, \ldots, i_b\}$
  for *nodes* $p = 0, \ldots, (k\text{-}1)$ *in parallel* **do**
    assign samples $\mathcal{D}_t^{(n)} = \{i_{p+1}, \ldots, i_{(p+1)b/k}\}$
    obtain: $\mathbf{g}_t^{(i)} := \nabla f_i(\mathbf{x}_t),\ \forall i \in \mathcal{D}_t^{(p)}$ (back-propagation)
    $\tilde{\mathbf{g}}_t^{(p)} := \frac{k}{b} \sum_{i=1}^{b/k} \mathbf{g}_t^{(i)}$ (local gradient aggregation)
    communicate local gradient update $\tilde{\mathbf{g}}_t^{(p)}$
  end
  $\tilde{\mathbf{g}}_t := \frac{1}{k} \sum_{p=0}^{k-1} \tilde{\mathbf{g}}_t^{(p)}$ (gather all gradient updates)
  $\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}_t$ (parameter update)
end



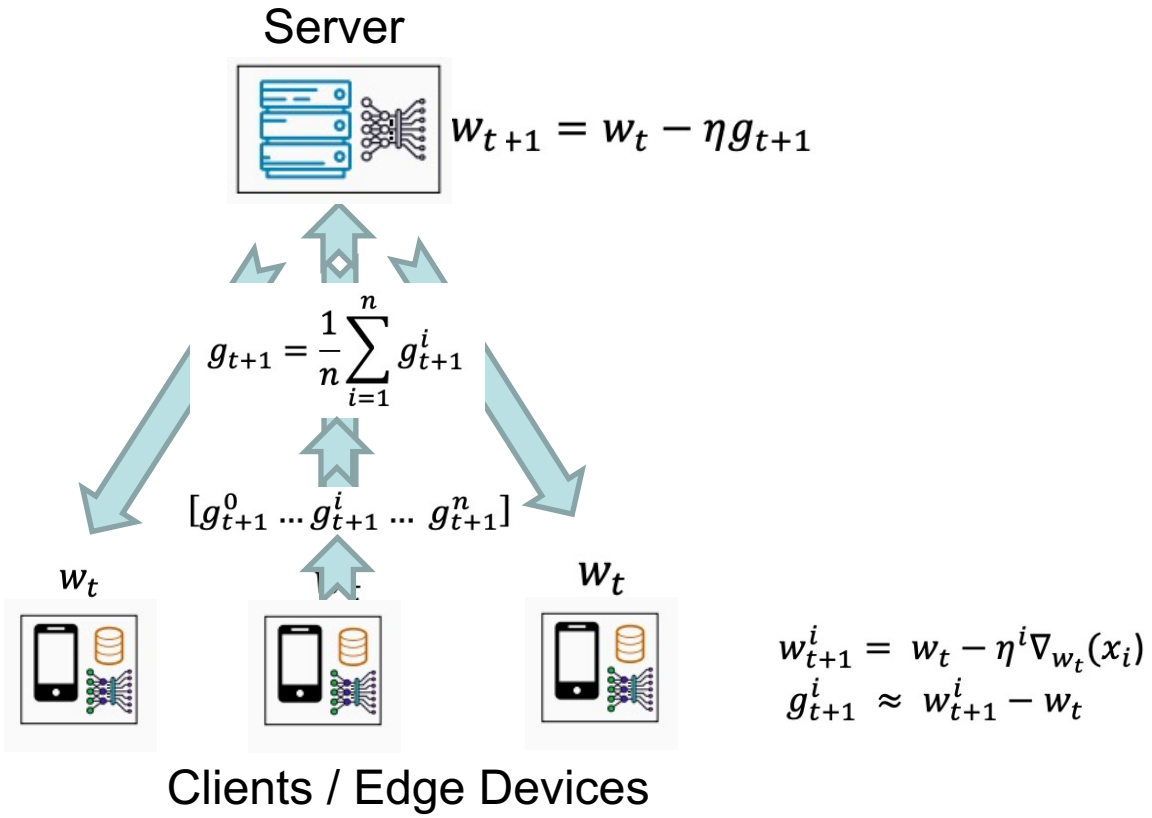
source: d2l.ai

A new paradigm – Federated Learning

- collaboratively train a ML model
- keep the data decentralized
- synchronous update scheme that proceeds in rounds of communication

McMahan, H. Brendan, Eider Moore, Daniel Ramage, and Seth Hampson. "Communication-efficient learning of deep networks from decentralized data." AISTATS, 2017

Federated Averaging



Vulnerability of Distributed Training

- **Byzantine Client**

  Byzantine failure in a distributed ML system occurs when one of the components in such a system behaves arbitrarily

  e.g. This could be the result of software bugs, hardware defects, communication loss (information dropout)

- **Adversarial Attack**

  Goal of the attacker is to simultaneously train on the main task and backdoor task by manipulating a subset of clients through adversarial trigger injection.

  e.g. force a word predictor to complete certain sentences with an attacker-chosen word, have search engine recommend adversary chosen product for a given query.

Recall : SGD

- Smooth non-convex problems with finite sum structure:

- Smooth non-convex problems with finite sum structure:
- SGD proceeds as follows:
- *Gross Corruption* Given b samples an adversary can replace $0 \leq \psi \leq 1/2$ fraction of them with arbitrary points. Suppose G and B are sets of good and bad points $\alpha = \frac{|B|}{|G|} = \frac{\psi}{\psi - 1} \leq 1$ (α corruption) and we want to solve:

- SGD proceeds as follows: $\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \tilde{\mathbf{g}}^{(t)}, \quad \tilde{\mathbf{g}}^{(t)} = \frac{1}{|\mathcal{D}_t|} \sum_{i \in \mathcal{D}_t} \mathbf{g}_i^{(t)}$

- *Gross Corruption* Given b samples an adversary can replace $0 \leq \psi \leq 1/2$ fraction of them with arbitrary points. Suppose G and B are sets of good and bad points $\alpha = \frac{|B|}{|G|} = \frac{\psi}{\psi - 1} \leq 1$ (α corruption) and we want to solve:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{|\mathbb{G}|} \sum_{i \in \mathbb{G}} f_i(\mathbf{x}) \right]$$

Corruption Model

- *Gross Corruption* Given b samples an adversary can replace $0 \leq \psi \leq 1/2$ fraction of them with arbitrary points. Suppose G and B are sets of good and bad points $\alpha = \frac{|B|}{|G|} = \frac{\psi}{\psi - 1} \leq 1$ (α corruption) and we want to solve:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{|\mathbb{G}|} \sum_{i \in \mathbb{G}} f_i(\mathbf{x}) \right]$$

- This corruption model covers a wide variety (if not all) of corruption :
  - Corruption in Feature
  - Corruption in Gradients
  - Corruption in Labels

Gradient Aggregation

- Given : b gradient estimates

$$\mathcal{D}_t = \{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\}$$

- When ψ = 0 : empirical mean is common measure of center.

$$\textsc{Mean}(\{g_t^i \in \mathbb{R}^d : \forall i \in [b]\}) := \mu = \frac{1}{b}\sum_{i\in[b]} g_t^i$$

- MSE minimization: $\mu = \arg\min_{y\in\mathbb{R}} \sum_{i=1}^{b} \|g_t^i - y\|^2$

Robust Gradient Aggregation

- **Breakdown Point :** smallest fraction of contamination that must be introduced to cause an estimator to break i.e. produce arbitrarily wrong estimates. Finite Sample breakdown point $1/b \leq \epsilon^* < 1/2$.

- No linear gradient aggregation (e.g. mean in SGD) strategy can tolerate even a single such corrupted point. Consider the following sample:

$$g_t^j = -\sum_{i \in \mathcal{D}_t \setminus j} g_t^i$$

- Mean has finite sample breakdown point of 1/b i.e. asymptotic breakdown 0

- Make SGD Robust: Replace Mean with Robust Mean Estimator.

Univariate Robust Gradient Aggregation

- (**Univariate Setting**) median is a measure which is robust to outliers. In fact, median achieves the optimal breakdown point of ½

- Given samples $\mathcal{D}_t = \{g_t^i \in \mathbb{R} : \forall i \in [b]\}$ median is the (b+1)/2 th ordered statistic if b is odd else is the mean of (b+1)/2 th and b/2 th ordered statistic.

- Median is also minimizer of sum of absolute errors:

$$\textsc{Med}(g_t^i : \forall i \in [b]) := \arg\min_{y \in \mathbb{R}} \sum_{i=1}^{b} |g_t^i - y|$$

Multivariate Robust Gradient Aggregation

- Coordinate wise Median (CM)

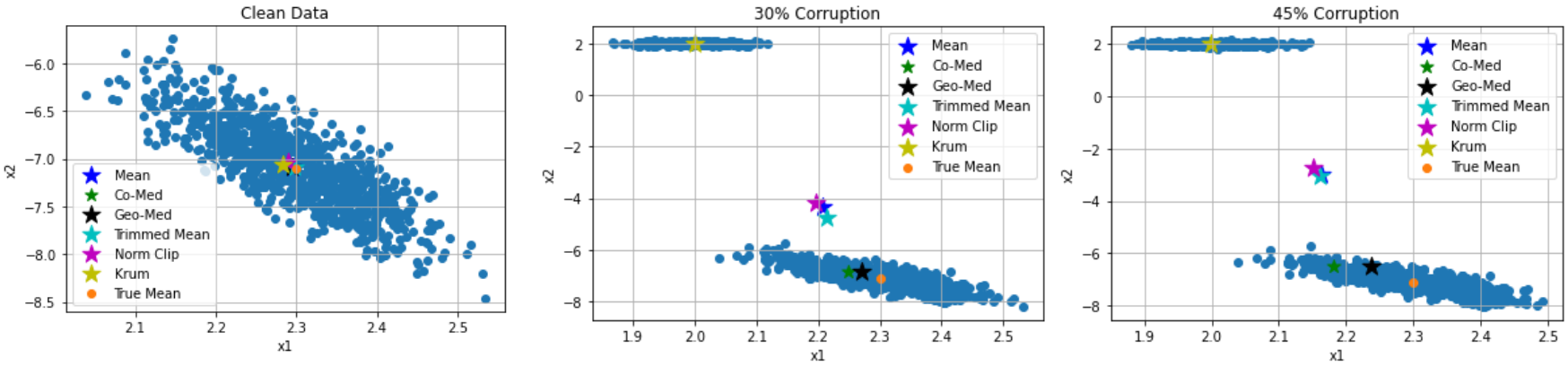
$$\textsc{Cm}(\{\mathbf{g}_t^i \in \mathbb{R}^d : \forall i \in [b]\})[k] = \textsc{Med}(\mathbf{g}_t^i[k] : \forall i \in [b])\ \forall k \in [d]$$

  - When d > 2 CM need not lie in the convex hull of the samples and are not orthogonal equivariant i.e. do not commute with co-ordinate transform

- Geometric Median (GM)

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

Multivariate Robust Gradient Aggregation



This Toy example in 2 dimensions demonstrates
the superior robustness properties of GM for estimating the aggregated gradient in presence of heavy corruption.

Robust SGD in High Dimension

Algorithm	Aggregation Operator*	Iteration Complexity**	Breakdown Point**
SGD	$\textsc{Mean}(\cdot)$	$\boldsymbol{\mathcal{O}(bd)}$	1/b
(Yang et al., 2019; Yin et al., 2018)	$\textsc{Cm}(\cdot)$	$\mathcal{O}(bd \log b)$	**1/2**
(Wu et al., 2020)	$\textsc{Gm}(\cdot)$	$\mathcal{O}(d\epsilon^{-2} + bd)$	**1/2**
BGMD (This work)	$\textsc{BGm}(\cdot)$	$\mathcal{O}(k\epsilon^{-2} + bd)$	**1/2**
(Data and Diggavi, 2020)	(Steinhardt et al., 2017)	$\mathcal{O}(db^2 \min(d, b) + bd)$	1/4
(Blanchard et al., 2017)	$\textsc{Krum}(\cdot)$	$\mathcal{O}(b^2 d)$	$\lfloor \beta \rfloor$
(Yin et al., 2018)	$\textsc{CTm}_\beta(\cdot)$	$\mathcal{O}(bd(1-2\beta) + bd \log b)$	$\lfloor \beta \rfloor$
(Ghosh et al., 2019; Gupta et al., 2020)	$\textsc{Nc}_\beta(\cdot)$	$\mathcal{O}(bd(2-\beta) + b \log b)$	1/b

Table 1: Comparison of time complexity and robustness properties of different robust optimization methods (also see Fig. 6) without any distributional assumptions on the data. The bold quantities show a method achieves the theoretical limits. The first four methods are related to robust aggregation based approaches while the last four are filtering based approaches. $*$ $\textsc{Cm}(\cdot)$ co-ordinate wise median, $\textsc{Gm}(\cdot)$ Geometric (spatial) median, $\textsc{BGm}(\cdot)$ Block Geometric Median, $\textsc{CTm}_\beta(\cdot)$ Co-ordinate wise Trimmed mean, $\textsc{Nc}_\beta(\cdot)$ Norm Clipping. $**$ In section A.2 we discuss the breakdown points and iteration complexities of these methods in more detail.

GM in High Dimension

- GM-SGD[1] : $x_{t+1} = \mathrm{x_t} - \eta \hat{g}_t \quad \hat{g}_t = \mathrm{GM}(\{\mathrm{x_i}\})$
- Unfortunately, finding GM is computationally hard.
- Best known algorithm[2] to find ϵ approximate GM of n points in R^d requires $O(d/\epsilon^2)$.
- GM is computationally intractable for optimization in high dimensions arising from deep learning models e.g. $d \approx$ 60M Alexnet, $d \approx$ 175B GPT3

1. Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 4613–4623,2018

2. Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In Proceedings of the forty-eighth annual ACM symposium on Theory of Computing, pages 9–21, 2016.

GM over low dimensional subspace

- Gradient distribution is often long tailed especially in case of over-parameterized deep learning settings.

- Intuition: Performing gradient aggregation in a low dimensional subspace might have little impact in the downstream optimization task.

- **BGMD** (proposed) judiciously subsets a block of k dimensions ($k << d$) and performs GM in R^k (Algorithm 1). Note this strategy is biased. BGMD introduces a memory mechanism to fix this bias and maintain same non-asymptotic convergence rate as SGD with GM in R^d.

Block coordinate GM Descent

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: $\text{GM}(\cdot)$

for *epochs* $t = 0, \ldots,$ *until convergence* **do**

- select samples $\mathcal{D}_t = \{i_1, \ldots, i_b\}$
- 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)
- $\boldsymbol{\Delta}_t := \mathcal{C}_k(\mathbf{G}_t) \in \mathbb{R}^{b \times k}$ (subset k dim via Algo. 2)
- $\mathbf{M}_{t+1} = \mathbf{G}_t - \boldsymbol{\Delta}_t$ (compute residuals)
- $\hat{\mathbf{m}}_{t+1} = \frac{1}{b} \sum_{0 \le i \le b} \mathbf{M}_{t+1}[i,:]$ (update memory)
- $\tilde{\mathbf{g}}_t := \text{GM}(\boldsymbol{\Delta}_t)$ (robust aggregation in $\mathbb{R}^k$)
- $\mathbf{x}_{t+1} := \mathbf{x}_t - \tilde{\mathbf{g}}_t$ (parameter update)

end

Algorithm 2 Block Coordinate Selection Strategy

Input: $\mathbf{G}_t \in \mathbb{R}^{n \times d},\ k$

for *coordinates* $j = 0, \ldots, d\text{-}1$ **do**

- $s_j \leftarrow \|\mathbf{G}_t[:,j]\|^2$ (norm along each dimension)

end

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: $\mathcal{C}_k(\mathbf{G}_t)$

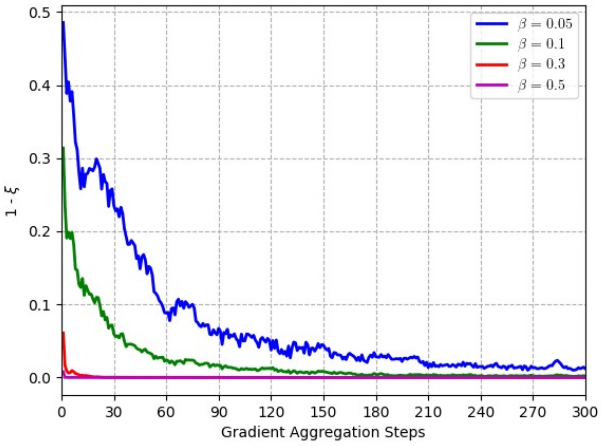
Block Selection Strategy

- what would be the best strategy to select the most informative block of coordinates?
  - Ideally, pick k dimensions that would result in the largest decrease in training loss. This is NP Hard ☹
  - Consider $\mathbf{G}_t \in \mathbb{R}^{b \times d}$ where each row is the transpose of one stochastic gradient estimate $\mathbf{G}_t[i,:] = (\mathbf{g}_i^{(t)})^T \in \mathbb{R}^{1 \times d},\ \forall i \in [b]$
  - Selecting k dimensions is equivalent to k column subset selection of the jacobian. We use active norm sampling* to pick k columns I_k of the jacobian (Algorithm 2)

$$\mathcal{C}_k(\mathbf{G}_t)[i, j \in \mathbb{I}_k] = \mathbf{G}_t[i,j],\ \mathcal{C}_k(\mathbf{G}_t)[i, j \notin \mathbb{I}_k] = 0$$

Block Selection Strategy

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



It is worth noting that without additional distributional assumption on the gradients the lower bound on ξ cannot be improved.

Note the worst case when all the gradients are uniformly distributed along each coordinate

In the figure we plot relative residual error
$\|\mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t)\|^2 / \|\mathbf{G}_t\|^2$ for training LeNet on Fashion MNIST

Memory

- Aggressively small k (which we want) implies large information loss , because restricting to k dimensions results in $\frac{d}{k}$ factor increase in variance.

- Solution: Keep track of Residual $M_t = ||G_t - C_k(G_t)|| \in R^{b \times d}$ (b is batch size, d is dimension) , initialized to 0.

- At each update step, accumulate residual error incurred by ignoring (d - k) dimensions, averaged overall the samples participating in that round referred as memory. In the next iterations add back memory to new gradient estimates.

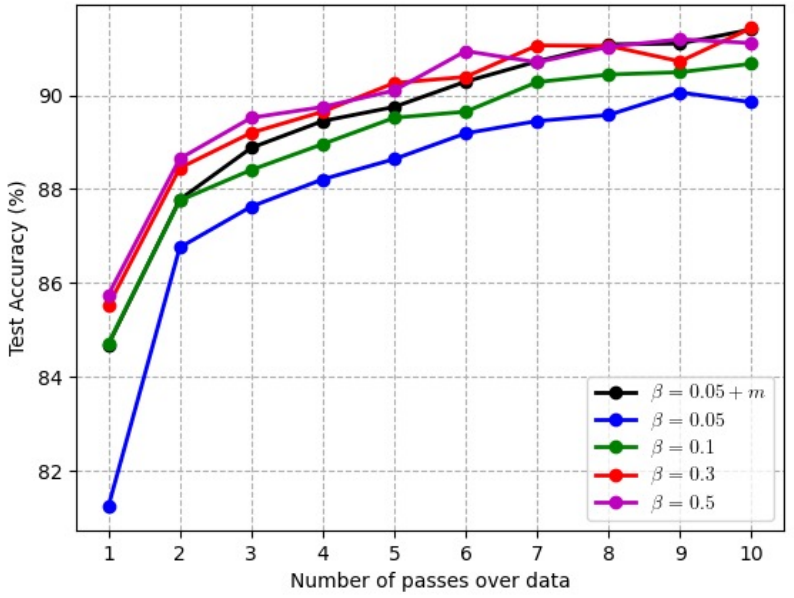
$$\mathbf{G}_t[i,:] = \gamma \mathbf{G}_t[i,:] + \hat{\mathbf{m}}_t \ \forall i \in [0, b]$$

$$\mathbf{M}_t = \mathbf{G}_t - \mathcal{C}_k(\mathbf{G}_t) \ ; \hat{\mathbf{m}}_{t+1} = \frac{1}{b} \sum_{0 \leq i \leq b} \mathbf{M}_t[i,:]$$

- Fixes the bias due to sampling and retains convergence

Benefit of Memory

- Allows to more aggressively reduce block size while preserving convergence.
- Additional overhead to update Memory is negligible compared to savings by aggressive block size reduction
- $k = \beta d\ ; 0 < \ \beta \leq 1$ We train LeNet on Fashion MNIST for different β. We see that training with the memory mechanism (m) enjoys the same accuracy while using a much smaller β

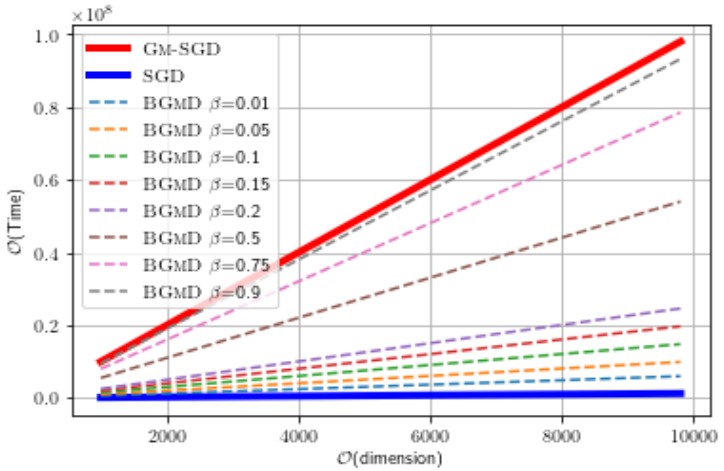


Computational Complexity

- Consider solving optimization problem with finite sum structure with parameters $\in R^d$ and batch size b using SGD like iterations.

Proposition 1. (Computational Complexity). *Given an ϵ- approximate* GM *oracle , each gradient aggregation of* BGMD *with block size k incurs a computational cost of: $\mathcal{O}(\frac{k}{\epsilon^2} + bd)$.*

Lemma 2. *Let $k \leq \mathcal{O}(\frac{1}{F} - b\epsilon^2) \cdot d$. Then, given an ϵ- approximate* GM *oracle, Algorithm 1 achieves a factor F speedup over* GM-SGD *for aggregating b samples.*



Computational Upper Bound

Theory: Assumptions

Let $f := 1/\mathbb{G} \sum_{i \in \mathbb{G}} f_i(\mathbf{x})$ denote the average of non-corrupt functions. Then, we also assume that the unconstrained problem $\arg\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$ has a non empty solution set $\mathcal{X}^*$. We will denote the optimal function value as $f(\mathbf{x}^*)$ where $\mathbf{x}^* \in \mathcal{X}^*$ and initial parameters by $\mathbf{x}_0$. For notational convenience define $R_0 = f(\mathbf{x}_0) - f(\mathbf{x}^*)$.

Assumption 1 (Stochastic Oracle). *Each non-corrupt sample $i \in \mathbb{G}$ is endowed with an unbiased stochastic first-order oracle with bounded variance, i.e.*

$$\mathbb{E}_{z \sim \mathcal{D}_i}[\mathbf{g}_i(\mathbf{x}, z)] = \nabla f_i(\mathbf{x}) \tag{8}$$

$$\mathbb{E}_{z \sim \mathcal{D}_i} \|\nabla F_i(\mathbf{x}, z)\|^2 \leq \sigma^2 \tag{9}$$

Assumption 2 (Smoothness). *Each non-corrupt function f_i is L-smooth i.e. $\forall i \in \mathbb{G}$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$:*

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}) + \langle \mathbf{x} - \mathbf{y}, \nabla f_i(\mathbf{y}) \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2 \tag{10}$$

Note that if f_i are twice differentiable then this implies that the eigenvalues of $\nabla^2 f_i(\mathbf{x})$ are bounded above by L.

Assumption 3 (Polyak-Łojasiewicz Condition). *f satisfies the Polyak-Łojasiewicz condition (PLC) with parameter $\mu > 0$ (Polyak, 1963; Karimi et al., 2016):*

$$\|\nabla f(\mathbf{x})\|^2 \geq 2\mu (f(\mathbf{x}) - f(\mathbf{x}^*)), \ \mu > 0 \tag{11}$$

Note that, PLC implies that every stationary point is a global minima but doesn't imply uniqueness and is a much milder condition than strong convexity (Karimi et al., 2016).

We now analyze the convergence properties of BGMD (Algorithm 1) and state the results in Theorem 1 and Theorem 2 for general non-convex functions and functions satisfying PLC, respectively. [7]

Convergence Guarantees

- Non-convex and Smooth : Suppose f_i corresponding to non-corrupt samples i.e. $i \in G$ are L smooth and non-convex. Run BGMD with ϵ approximate GM oracle with $\gamma = \frac{1}{2L}$ in presence of α corruption for T iterations. Sample any iteration τ uniformly at random then:

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

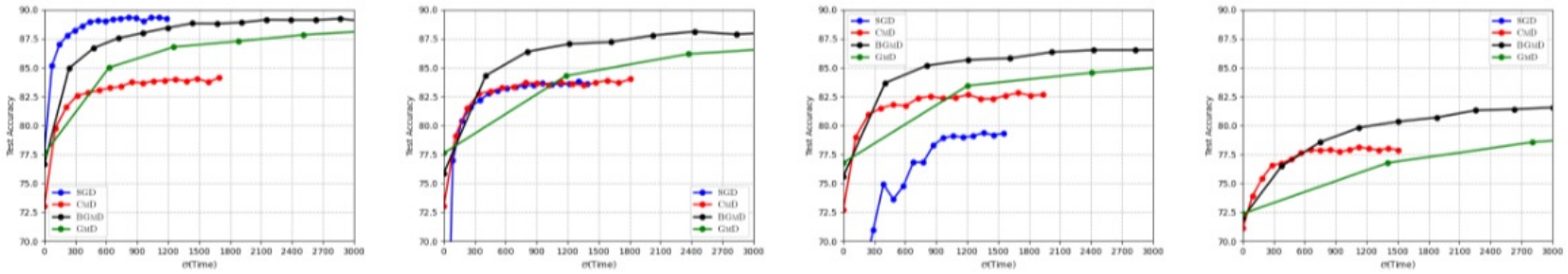
- Non Convex PLC : Suppose f_i further satisfies PLC with parameter μ then running BGMD with $\gamma = \frac{1}{4L}$ satisfies:

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

Remark 1 (BGMD Breakdown Point). BGMD *converges to the neighborhood of a stationary point* $\forall 0 \leq \psi < 1/2$ *i.e. has optimal breakdown point of 1/2.*

Empirical Evidence: Feature Corruption

- Feature Corruption
  - Additive Corruption (Huber's Contamination):
    $z_t \sim \mathcal{N}(100, 1)$ directly added to the image.
  - Impulse Corruption:
    Salt and Pepper noise added by setting 90% of pixels to 0 or 1.



(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 2: **Robustness to Feature Corruption**: Test accuracy of different schemes as a function of **wall clock time** for training Fashion-MNIST using LeNet (i.i.d) in presence of **impulse noise**. Observe that BGMD is able to maintain high accuracy even in presence of strong corruption while attaining at least 3x speedup over GMD whereas CMD performs sub-optimally and SGD diverges at such levels of corruption. Further, note that in clean setting, BGMD can almost reach the same accuracy of SGD while using the same compute budget. * Note that all the algorithms were run for same number of epochs.

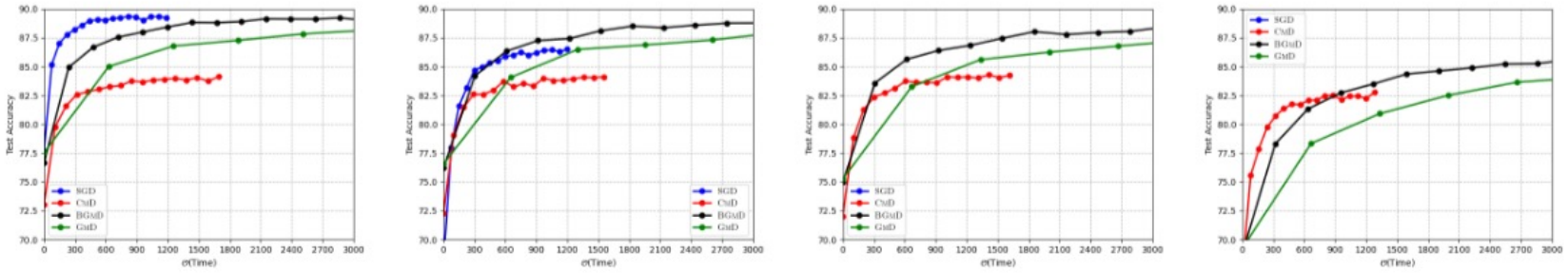
Empirical Evidence: Gradient Corruption

- Gradient Corruption
  - Additive Corruption (Huber's Contamination):

$$g_t^c = g_t + z_t \quad where \quad z_t \sim \mathcal{N}(0, 100)$$

  - Bit Flip Corruption: scaled bit flipped version of the actual gradient

$$g_t^c = -100\, g_t$$

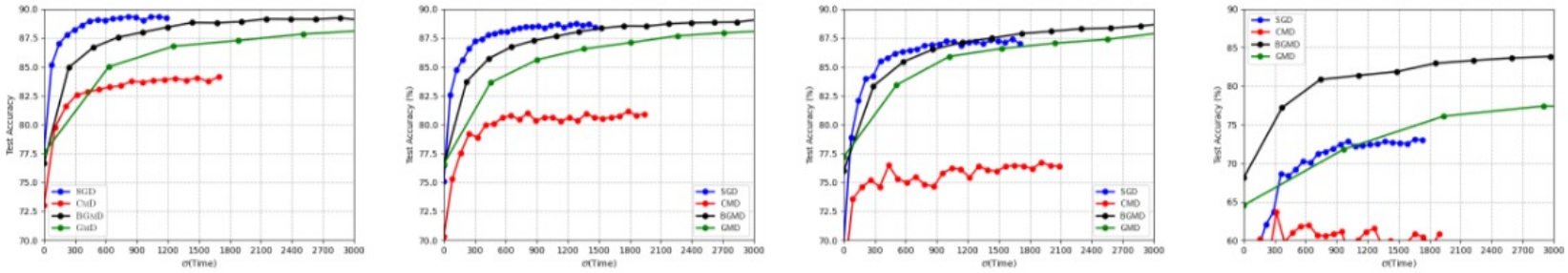


(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 3: **Robustness to Gradient Corruption**: Training Fashion-MNIST using LeNet in i.i.d setting in presence of **scaled bit flip corruption** to stochastic gradients. Similar to Figure 2, BGMD remains highly robust. Further, as seen from the plots against **wall clock time** BGMD results in more than 2.5x speedup over all settings. Further in clean setting, BGMD can almost reach the same accuracy of SGD while using the same compute budget.

Empirical Evidence: Label Corruption

- Label Corruption
  - Backdoor Attack: flip the labels of randomly chosen ψ fraction of samples to a target label.



(a) No corruption (b) 10% Corruption (c) 20% Corruption (d) 40% Corruption

Figure 4: **Robustness to Label Corruption**: Training Fashion-MNIST (iid) with LeNet in presence of **backdoor attack**. We note similar superior performance of BGMD while resulting in more than 2.5x speedup over GMD.

Empirical Evidence: Generalization

	Corruption (%)	SGD	CMD	BGMD	GMD
LeNet - Fashion MNIST (homogeneous)					
Clean	-	**89.39**±0.28	83.82±0.26	89.25±0.19	88.98±0.3
Gradient Corruption					
Bit Flip	20	-	84.20±0.02	**88.42**±0.16	88.07±0.05
	40	-	82.33±1.60	**85.67**±0.09	85.57±0.09
Additive	20	-	72.55±0.16	**87.87**±0.33	87.24±0.16
	40	-	41.04±1.13	**88.29**±0.01	83.89±0.08
Feature Corruption					
Additive	20	-	82.38±0.13	**86.76**±0.03	86.63±0.04
	40	-	78.54±0.65	**82.27**±0.06	81.23±0.03
Impulse	20	79.18±6.47	82.59±0.60	**86.91**±0.36	86.23±0.03
	40	-	78.03±0.73	**82.11**±0.73	81.41±0.12
Label Corruption					
Backdoor	20	86.99±0.02	76.38±0.13	**88.97**±0.10	88.26±0.04
	40	73.01±0.68	60.85±1.24	**84.69**±0.31	81.32±0.16
ResNet18 - CIFAR10 (heterogeneous)					
Clean	-	82.29±1.32	85.50±1.43	84.82±0.76	**85.65**±0.48
Gradient Corruption					
Bit Flip	20	-	80.87±0.21	87.56±0.06	**88.07**±0.05
	40	-	77.41±1.04	**82.66**±0.31	80.81±0.01
Additive	20	20.7±1.56	54.75±0.38	**83.84**±0.12	82.40±0.90
	40	-	23.35±6.13	**82.79**±0.68	79.46±0.24

Table 2: Summary of generalization performance under variety of corruption settings. Missing values (-) denotes that the training has diverged. It is clear, that in addition to being efficient BGMD also enjoys superior generalization performance. While, this is an interesting future work, it is possible that the resulting jacobian compression operator $\mathcal{C}_k(\cdot)$ via Algorithm 2 results in implicit regularization benefits in high dimensional settings (Anonymous, 2022; Gower et al., 2020; Wu et al., 2019) explaining the superior performance.

Empirical Evidence: Generalization

Table 3: 1.12M parameter CNN trained on MNIST in regular i.i.d. setting. For all corruption types, test accuracy of BGMD is similar to that of GMD and surprisingly, in some cases even higher. As expected, SGD fails to make progress under corruption . CMD performs sub-optimally as corruption is increased.

	Corruption (%)	SGD	CMD	BGMD	GMD
Clean	-	**99.27**±0.01	98.83±0.02	99.09±0.05	99.24±0.02
		Gradient Attack			
Bit Flip	20	9.51±1.77	98.79±0.01	**99.06**±0.02	98.98±0.01
	40	9.60±2.04	93.69±0.09	97.89±0.05	**98.11**±0.12
Additive	20	9.68±0.11	94.26±0.03	98.61±0.01	**98.69**±0.01
	40	9.74±0.12	91.86±0.03	**97.78**±0.27	92.78±0.04

Extension: QBGMD

- In distributed setting, gradient updates are often compressed to reduce communication cost.

Lemma 4. *Let $\mathcal{C}_k : \mathbb{R}^d \to \mathbb{R}^d$ be coordinated sparse approximation operator as described in Algorithm 2 and $Q_s : \mathbb{R}^d \to \mathbb{R}^d$ be a randomized quantization operator, then $\mathcal{C} := \mathcal{C}_k(Q_s(\mathbf{x}))$ is also a contractive compression with $\delta = (1 - \beta_{k,s})\frac{k}{d}$ i.e. for every $\mathbf{x} \in \mathbb{R}^d$:*

$$\mathbf{E}_{\mathcal{C}_k, Q_s}[\|\mathbf{x} - \mathcal{C}(\mathbf{x})\|^2] \leq \left[1 - (1 - \beta_{k,s})\frac{k}{d}\right] \|\mathbf{x}\|^2$$

- Now, this lemma immediately gives a communication efficient version of our algorithm referred as Quantized BGMD (QBGMD) where the clients communicate only quantized gradients.

- The convergence rates are identical i.e. Theorem 1,2 hold with: $0 < \xi \leq \left(1 - \beta_{k,s}\right)\frac{k}{d}$)

Extension: FedBGMD

Algorithm 3 FL-BGMD

Input: stepsize γ, number of iterations T, synchronization rounds $\mathcal{I}_T$, accuracy of the GM oracle $\{\epsilon\}_{t=0}^{T-1}$;
initialize: $\mathbf{x}_0 = \mathbf{x}_0^i = \mathbf{y}_0^i$, $\mathbf{m}_0^i = \mathbf{0}$ for all $i \in [n] \backslash \mathbb{B}$
for $t = 0, \ldots, T-1$ **do**
    On Clients:
    for $i = 1, \ldots, n$ in parallel **do**
        if $i \in \mathbb{G}$ **then**
            $\mathbf{g}_t^i = \nabla F_i(\mathbf{y}_t^i, z_t^i)$ (computing the local stochastic gradient $\mathbf{g}_t^i$)
            $\mathbf{y}_{t+0.5}^i = \mathbf{y}_t^i - \gamma \mathbf{g}_t^i$ (local first-order update)
            if $t+1 \notin \mathcal{I}_T$ **then**
                $\mathbf{x}_{t+1} = \mathbf{x}_t, \mathbf{y}_{t+1}^i = \mathbf{y}_{t+0.5}^i$ (updating only the local model)
            else
                $\mathbf{c}_t^i = \mathcal{Q}(\mathbf{x}_t - \mathbf{y}_{t+0.5}^i)$ (sending the message $\mathbf{c}_t^i$ to the server)
                $\mathbf{y}_{t+1}^i = \mathbf{x}_{t+1}$ (receive the aggregated model from the server)
            end if
        else
            $\mathbf{g}_t^i = \square, \quad \mathbf{c}_t^i = \blacksquare, \quad \forall i \in \mathbb{B}$ (arbitrary messages)
        end if
    end for
    At Server:
    if $t+1 \notin \mathcal{I}_T$ **then**
        $\mathbf{x}_{t+1} = \mathbf{x}_t$, (no updates to the global model)
    else
        $\mathbf{G}_t \in \mathbb{R}^{n \times d}$ s.t. $\mathbf{G}[i,:] = \mathbf{c}_t^i$
        $\mathbf{P}_t := \gamma_t \mathbf{G}_t + \mathbf{M}_t^s$ (server error correction)
        $\mathbf{\Delta}_t := \mathcal{C}_k(\mathbf{G}_t)$ (Run Algorithm 2)
        $\mathbf{M}_{t+1}^s := \mathbf{P}_t - \mathbf{\Delta}_t$ (update server residual)
        $\tilde{\mathbf{g}}_t := \text{GM}(\epsilon, \mathbf{\Delta}_t)$ (Robust Aggregation in $\mathbb{R}^k$)
        $\mathbf{x}_{t+1} := \mathbf{x}_t - \tilde{\mathbf{g}}_t$ (Global model update)
    end if
end for

Convergence Guarantee FedBGMD

Theorem 3 (Non-convex). *Consider the general case where the functions f_i corresponding to non-corrupt samples $i \in \mathbb{G}$ in (16) are **non-convex** and **smooth** (Assumption 2). Define, $R_0 := f(\mathbf{x}_0) - f(\mathbf{x}^*)$ where $\mathbf{x}^*$ is the true optima and $\mathbf{x}_0$ is the initial parameters. Run Algorithm 3 with compression factor $(1-\beta_{k,s})\frac{k}{d} \leq \xi \leq 1$ (Lemma 5), learning rate $\gamma = 1/2L$ and $\epsilon-$approximate $\textsc{Gm}(\cdot)$ oracle in presence of $\alpha-$corruption (Definition 1) for T iterations. Then the iterates satisfy:*

$$\frac{1}{T|\mathbb{G}|}\sum_{t=0}^{T-1}\sum_{i\in\mathbb{G}}\mathbb{E}\|\nabla f(\mathbf{y}_t^i)\|^2 \leq \frac{8R_0}{\gamma T} + 8L\gamma\sigma^2 + 24L^2\gamma^2H^2\sigma^2\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{2304|\mathcal{I}_T|H^2\sigma^2}{T(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{48|\mathcal{I}_T|\epsilon^2}{\gamma^2T|\mathbb{G}|^2(1-\alpha)^2} \tag{17}$$

Theorem 4 (Non-convex under PLC). *Consider that functions f_i in (16) are **non-convex** and also satisfies the **Polyak-Łojasiewicz Condition** (Assumption 3) with parameter μ. Define, $R_0 := f(\mathbf{x}_0) - f(\mathbf{x}^*)$ where $\mathbf{x}^*$ is the true optima and $\mathbf{x}_0$ is the initial parameters. After T iterations Algorithm 3 with compression factor $(1-\beta_{k,s})\frac{k}{d} \leq \xi \leq 1$ (Lemma 5), learning rate $\gamma = 1/4L$ and $\epsilon-$approximate $\textsc{Gm}(\cdot)$ oracle in presence of $\alpha-$corruption (Definition 1) satisfies:*

$$\frac{1}{|\mathbb{G}|}\sum_{i\in\mathbb{G}}\mathbb{E}\|\hat{\mathbf{y}}_T^i - \mathbf{x}^*\|^2 \leq \frac{16(f(\mathbf{x}_0)-f^*)}{\mu^2\gamma}\left[1-\frac{\mu\gamma}{2}\right]^T + \frac{16L\gamma\sigma^2}{\mu^2} + \frac{40L^2}{\mu^2}\gamma^2H^2\sigma^2\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{3072H^2\sigma^2}{\mu^2(1-\alpha)^2}\left[1+\frac{4(1-\xi^2)}{\xi^2}\right] + \frac{64\epsilon^2}{\mu^2\gamma^2|\mathbb{G}|^2(1-\alpha)^2}, \tag{18}$$

for a global optimal solution $\mathbf{x}^* \in \mathcal{X}^*$. Here, $\hat{\mathbf{y}}_T^i := \frac{1}{W_T}\sum_{t=0}^{T-1} w_t\mathbf{y}_t^i$ with weights given as $w_t := (1-\frac{\mu\gamma}{2})^{-(t+1)}$, $W_T := \sum_{t=0}^{T-1} w_t$.