紹介論文

Byzantine Stochastic Gradient Descent

Alistarh, Dan and Allen-Zhu, Zeyuan and Li, Jerry

In *Proceedings of Advances in Neural Information Processing Systems 31*, 2018, pages 4614–4624 (NIPS2018)

Abstract

本論文では, 各イテレーション毎に確率的勾配法 (Stochastic Gradient Descent, SGD) を行う m 個の計算機のうちのある割合が敵対的に振る舞う可能性があるビザンチンである状況を仮定し, 敵対的環境における分散確率的最適化を行う. 本論文の提案手法は, 凸関数に関して, $T = \tilde{O}\left(\frac{1}{\epsilon^2 m} + \frac{\alpha^2}{\epsilon^2}\right)$ イテレーション必要とする. 一方で, 標準的なミニバッチ SGD では $T = O\left(\frac{1}{\epsilon^2 m}\right)$ イテレーション必要とする. これは, 提案手法よりも少ないイテレーションであるが, ビザンチン障害に対して堅牢な方法ではない. 下界を示し, 提案手法が情報理論的に最適であることを示す.

1 Introduction

データが元々バラバラな場所にある (フェデレーションラーニング [1]), もしくは, 並列的な計算をするためにデータを複数の計算機に分割されているために, 機械学習アプリケーションはますます分散化が進んでいる. このような分散設定では, 堅牢性が非常にとなってくる. 例えば, 一部の計算機がクラッシュする, または予期しない動作をする可能性がある. しかし, 一部の計算機がそのような異常な動作をした場合でも, 全体としては, 正常に機能するようにふるまわなければならない. 一部の学習者 (ワーカー) が任意にふるまうビザンチン障害に関する研究は, 盛んに研究されている. また, ビザンチン障害に関する機械学習アルゴリズムの設計は, 比較的最近の話題であるが, 急速に研究されている.

本論文では, 分散学習におけるビザンチン障害に対する学習アルゴリズムを少ないデータから高い精度を要求する”sample complexity”と, 分散化によって達成される実行速度を損なわない計算の複雑さ”computation complexity”の 2 つの基準で評価する. さらに, これらの基準は高次元でも成立することが必要な条件である.

System Model

本論文では, ビザンチン障害における確率的最適化を行う. 関数 $\mathbb{R}^d \to \mathbb{R}$ 上の未知な分布 $\mathcal{D}$ を仮定し, $f(x) := \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]$ を最小化することを目標とする. m 人のワーカーとマスターが存在し, ワーカー内の割合 $\alpha < \frac{1}{2}$ がビザンチンワーカーだと仮定する. 各ワーカーは分布 $\mathcal{D}$ から T 個の関数を取得する. 本問題の構造は次のようになる : ワーカーはローカルな計算をして, 結果をマスターに送信する. マスターはワーカーから受け取った情報をもとに何らかの計算をして, ワーカーに計算結果を送る. マスターは, 最終的には, おおよそ f を最小化する値を出力するはずである.

本論文では, k 回目のイテレーションで, マスターは現在のパラメータ $x_k \in \mathbb{R}^d$ を各ワーカーに送信し, ワーカーは x_k での勾配を計算し, マスターに送る. 正常なワーカーは分布 $\mathcal{D}$ からランダムサンプリングされた f_s の勾配 ∇f_s を送るが, ビザンチンワーカーは任意の敵対的な値を送るかもしれない. 標準的な確率的最適化ではたった一人のビザンチンワーカーでさえも防ぐことができない.

”sample complexity”は, アクセスした関数 $f_s(\cdot)$ の数で決定される. 各ワーカーは各反復ごとに 1 つのサンプルを取得するために, ”sample complexity”を最小化することは反復の回数を最小化することと同等である.

”time complexity”も同様に反復回数によって決まる.

Our Results

本論文では, 損失関数が凸である場合のビザンチン確率的最適化問題を議論する. 各ワーカーの損失関数 $f_s(x)$ が凸でなくても, $f(x)$ は凸であると仮定する. ビザンチンワーカー存在下において, 対数やより低次の項までの収束を保証する.

(1). 最適な” sample complexity”を達成する
(2). 最適な”computation complexity”を達成する
(3). $\alpha \to 0$ の場合に最適な” sample complexity”と”time complexity を達成する
(4). データの次元数が大きくなった場合でも (1)〜(3) を達成する

それに加えて, 提案手法では, 敵対者の割合が半数未満 $\alpha < \frac{1}{2}$ の場合でも堅牢であることが保証される. 既存研究 ([2, 3, 4, 5, 6, 7, 8]) では, データが高次元である場合において, (1)〜(4) のいずれかを達成するアルゴリズムは存在しない. 既存研究のアルゴリズムは, 弱い堅牢性を保証するか, 次元数 d または, 誤差 ϵ とともに”sample complexity”または, ”time complexity”が多項的に劣化する.

Technical Contribution

本論文では, 各ワーカー $i \in [m]$ は各イテレーションで SGD を行う. $v_i^{(k)}$ を時刻 $k \in [T]$ のワーカー $i \in [m]$ による勾配とする. martingale concentration より, 正常なワーカー i による $B_i := \frac{v_i^1 + \cdots + v_i^{(T)}}{T}$ は $B_* := \frac{\nabla f(x_1) + \cdots + \nabla f(x_T)}{T}$ の付近に 誤差 $\frac{1}{\sqrt{T}}$ で集中するはずである (?). したがって, もし $\|B_i - B_*\| > \frac{1}{\sqrt{T}}$ ならば, i はビザンチンワーカーである.

しかし, 2 つの障害がある. まず, 新しいビザンチンワーカーを見つけるたびにアルゴリズムを再起動することはできない. 2 つ目は, ビザンチンワーカーは上記の基準に違反しないことでビザンチンワーカーであると認識されないことである. 最初の障害を解決するために各時刻 k において, 次の値を追跡する.

$$B_i^{(k)} := \frac{v_i^1 + \cdots + v_i^{(k)}}{k}$$

もし $B_*^{(k)}$ から離れすぎていたらビザンチンワーカーであるので, 取り除く. 2 つ目の障害を解決するために, 似たような基準を取り入れる.

$$A_i^{(k)} := \frac{\langle v_i^{(1)}, x_1 - x_0 \rangle + \cdots + \langle v_i^{(k)}, x_k - x_0 \rangle}{k}$$

正常なワーカーは両方の基準を満たすことを証明する. さらに, これらの両方を満たすビザンチンワーカーは, アルゴリズムの収束に無視できるほどの悪影響しか及ぼさない.

Related Work

[8] は本論文と最も関係のある研究である. しかし, 勾配降下法 (Gradient dicent, GD) によるビザンチン障害による堅牢性を考慮していた. 彼らの研究では, m 人のワーカーが n 個のデータを受け取る. 各イテレーション k において, ワーカー i は n 個のデータから勾配を計算して, それを平均することに x_k における勾配を計算する. そして, 受け取った m 個の勾配からメディアンやトリム平均による集約規則によって, 1 つの勾配を得る.

つまり, 合計 mnT 個の勾配を計算する. 一方で本論文で提案する Byzantine variant of SGD では, 計算する勾配の数は合計 Tm 個である.

”sample complexity”の観点では, 次元 d が大きくなるにつれ, アルゴリズムの複雑さが大きくなっていく. これは coordinate-wise な操作によるものが大きい. それにより, 高次元な場合, 既存手法では最適でなくなってしまう. また, 彼らは我々よりも弱い家庭で議論している. 彼らは勾配の確率的な誤差 $\nabla f_s(x) - \nabla f(x)$ の variance や skewness を仮定している. 一方で本論文では, $\nabla f_s(x) - \nabla f(x)$ は確率 1 で抑えられることのみを仮定している. その結果, 我々のより強い仮定はアルゴリズムや分析を簡単にする.

2 Preliminaries

ユークリッドノルムを $\|\cdot\|$ し, $[n] \coloneqq \{1, 2, \ldots, n\}$ とする. また, 強凸性, 平滑性, リプシッツ連続性を述べる.

Definition 2.1. For a differentiable function$f : \mathbb{R}^d \to \mathbb{R}$

- f is σ-strongly convex if $\forall x, y \in \mathbb{R}^d$, it satisfies $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma}{2}\|x - y\|^2$.
- f is L-Lipschitz smooth if $\forall x, y \in \mathbb{R}^d$, $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$.
- f is G-Lipschitz continuous if if $\forall x, \in \mathbb{R}^d$, $\|\nabla f(x)\| \leq G$.

Byzantine Convex Stochastic Optimization

m 人のワーカーとその中で最大でも割合 $\alpha \in [0, \frac{1}{2})$ のビザンチンワーカーが存在すると仮定する. また, good $\subset [m]$ を正常なワーカーの集合とする. $\mathcal{D}$ を関数 $f_s : \mathbb{R}^d \to \mathbb{R}$(必ずしも凸である必要はない) の分布とする. ただし, $f(x) \coloneqq \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]$ は凸であると仮定する. 本論文の目的は以下を最小化することである.

$$\min_{x\in\mathbb{R}^d} \{f(x) \coloneqq \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]\}. \tag{2.1}$$

さらに勾配について, 以下を仮定する.

Assumption 2.2. For each iteration $k \in [T]$ and for every $i \in$ good, we have $\nabla_{k,i} = \nabla f_s(x_k)$ for a random sample $\mathcal{D}$, and $\|\nabla_{k,i} - \nabla f(x_k)\| \leq \mathcal{V}$

また, 各時刻 $k \in [T]$ において, $i \notin$ good である勾配 $\nabla_{k,i}$ は敵対的に決めることができ, $\{\nabla_{k',i}\}_{k'\leq k, i\in[m]}$ に依存する可能性がある.

3 Description and Analysis of ByzantineSGD

一般性を失わずに, 初期値 $x_1 \in \mathcal{R}^D$ を仮定し, 以下を解決することが目標である.

$$\min_{\|x-x_1\|\leq D} \{f(x) \coloneqq \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]\}. \tag{3.1}$$

(3.1) を最小化する値を x^* とする.

本論文のアルゴリズムである BYzantineSGD を Algorithm 1 に定義した. 各イテレーション毎に x_k は, ByzantineSGD は good_{k-1} から正常なワーカーを特定しようとする. そして, good_k のみである $\xi_k \coloneqq \frac{1}{m}\sum_{i\in\text{good}_k} \nabla_{k,i}$ によって, パラメータ更新する.

各ワーカー $i \in [m]$ は値 $A_i = \sum_{t=1}^{k} \langle \nabla_{t,i}, x_t - x_1 \rangle$ とベクトル $B_i = \sum_{t=1}^{k} \nabla_{t,i}$ を計算する. A_{med} を $\{A_1, \ldots, A_m\}$ のメディアンとし, B_{med} を $\{B_1, \ldots, B_m\}$ のメディアンとする. また, ∇_{med} を $\{\nabla_{k,1}, \ldots, \nabla_{k,m}\}$ のメディアンとする. 本論文の ∇_{med} は次のように選択される. $|\{j \in [m] : \|\nabla_{k,j} - \nabla_{k,i}\| \leq 2\mathcal{V}\}| > \frac{m}{2}$ を満たす任意の $\nabla_{k,i}$ を ∇_{med} として選択する.

$\text{good}_0 = [m]$ から始める. また A_i は A_{med} の $\mathcal{L}_A$ だけ近傍にあり, B_i は B_{med} の $\mathcal{L}_B$ だけ近傍にあり, $\nabla_{k,i}$ は ∇_{med} の $4\mathcal{V}$ だけ近傍にあることを定義する. 閾値である $\mathcal{L}_A$ と $\mathcal{L}_B$ を適切に選択すれば, good_k は常に good な全てのワーカーを含むことを示していく.

Bounding the Error

ByzantineSGD によるエラーを以下ように定義する.

$$\text{Error}_1 \coloneqq \sum_{k \in [T]} \sum_{i \in \text{good}_k} \langle \nabla_{k,i} - \nabla f(x_k), x_k - x^* \rangle,$$

$$\text{Error}_2 \coloneqq \frac{1}{T} \sum_{k \in [T]} \| \frac{1}{m} \sum_{i \in \text{good}_k} (\nabla_{t,i} - \nabla f(x_k)) \|^2.$$

Error_1 は正常なワーカーによる SGD によって生み出されるバイアスとビザンチンワーカーによるノイズによるバイアスによって生み出されるものである. 一方で, Error_2 は $\nabla f(x_k)$ を近似するために ξ_k を用いることによるバリアンスである. Error_1, Error_2 に妥当な推定を行うために, $\{A_i\}_i$ と $\{B_i\}_i$ を使用して, 悪意のあるワーカーを排除する. アルゴリズムが終了しても, good_T に敵対者が含まれているかもしれないことに注意する必要がある. ただし, それらの敵対的なノイズはごく僅かでなければならず, アルゴリズムの性能に影響を与えてはならない. 2 つの誤差の範囲を確立するために, 次の補題を示す.

Lemma 3.1. With probability $1 - \delta$, we simultaneously have

$$|\text{Error}_1| \leq 4D\mathcal{V}\sqrt{TmC} + 16\alpha m D\mathcal{V}\sqrt{TC} \;\; \textit{and} \;\; \text{Error}_2 \leq 32\alpha^2\mathcal{V}^2 + \frac{4\mathcal{V}^2 C}{m}.$$

Smooth functions

目的関数が smooth である場合を考慮し, 収束率を導出する.

Theorem 3.2. Suppose in (3.1) our $f(x)$ is L-smooth and Assumption 2.2 holds. Suppose $\eta \leq \frac{1}{2L}$ and $\mathcal{L}_A = 4D\mathcal{V}\sqrt{TC}$ and $\mathcal{L}_B = 4\mathcal{V}\sqrt{TC}$. Then, with probability at least $1 - \delta$, letting $C \coloneqq \log(\frac{16mT}{\delta})$ and $\bar{x} \coloneqq \frac{x_2 + \cdots + x_{T+1}}{T}$, we have

$$f(\bar{x}) - f(x^*) \leq \frac{D^2}{\eta T} + \frac{8D\mathcal{V}\sqrt{TmC} + 32\alpha m D\mathcal{V}}{Tm} + \eta \cdot \left(\frac{8\mathcal{V}^2 C}{m} + 32\alpha^2\mathcal{V}^2 \right).$$

If η is chosen optimally, then

$$f(\bar{x}) - f(x^*) \leq O\left(\frac{LD^2}{T} + \frac{D\mathcal{V}\sqrt{C}}{\sqrt{Tm}} + \frac{\alpha D\mathcal{V}\sqrt{C}}{\sqrt{T}} \right).$$

- The first term $O\left(\frac{LD^2}{T}\right)$ is the classical error rate for gradient descent on smooth objectives and should exist even if $\mathcal{V} = 0$ (so every $\nabla_{k,i}$ exactly equals $\nabla f(x_k)$) and $\alpha = 0$.

- The first two terms $\widetilde{O}\left(\frac{LD^2}{T} + \frac{D\mathcal{V}}{\sqrt{Tm}}\right)$ together match the classical mini-batch error rate for SGD on smooth objectives, and should exist even if $\alpha = 0$ (so we have no Byzantine machines).
- The third term $\widetilde{O}\left(\frac{\alpha D\mathcal{V}}{\sqrt{T}}\right)$ is optimal in our Byzantine setting due to Theorem 4.3.

Nonsmooth functions

目的関数が non-smooth である場合も同様に厳密な収束率を導出した.(Theorem 3.3.)

Strongly convex functions

$$\min_{x\in\mathbb{R}^d} f(x) \coloneqq \mathbb{E}_{s\sim\mathcal{D}}[f_s(x)]. \qquad where\ f(x)\ is\ \sigma-strongly\ convex. \tag{3.6}$$

次に, 損失関数が強凸である場合のデータ数に関する収束を示す.

Theorem 3.4. Suppose in (3.6) our $f(x)$ is L-smooth and Assumption 2.2 holds. Given $x_0 \in \mathbb{R}^d$ with guarantee $\|x_0 - x^*\| \le D$, one can repeatedly ByzantineSGD to find a point x satisfying with probability at least $1-\delta'$, $f(x) - f(x^*) \le \epsilon$ and $\|x - x^*\| \le \frac{2\epsilon}{\sigma}$ in

$$T = \widetilde{O}\left(\frac{L}{\sigma} + \frac{\mathcal{V}^2}{m\sigma\epsilon} + \frac{\alpha^2\mathcal{V}^2}{\sigma\epsilon}\right)$$

iterations, where the $\tilde{O}$ notation hides logarithmic factors in $D, m, L, \mathcal{V}, \sigma^{-1}, \epsilon^{-1}, \delta^{-1}$.

目的関数が non-smooth である場合も同様に厳密な収束率を導出した.(Theorem 3.5.)

4 Lower Bounds for Byzantine Stochastic Optimization

5 Conclusion

ビザンチンワーカー存在下において, 悪意あるワーカーの新しい検出基準を設定し, タイトな収束率を求めた. この設定はビザンチン SGD の最も基礎的な設定であるが, 多くの未解決の問題が残っている. 今後の研究において 2 つの問題が考えられる. 1 つ目は, 正しく勾配を集約できるサーバが存在しない分散モデルである. 2 つ目は, 本論文のアルゴリズムの実用的な実装を図ることである. 本論文のアルゴリズムは伝統的なミニバッチ SGD に単純なチェックを追加するだけである. その結果, 最小のオーバーヘッドを追加すると同時に, 強力な堅牢性を保証する. また, 本論文のアルゴリズムはデータを各反復毎に取得するので, オンラインな状況を想定してる. これは, オフライン, つまりデータセット全体を見ることによる [8] のような以前のアプローチとは対照的である. 本論文の手法の主な利点は, 毎回全てのデータセットを調べる必要がないので, 各反復毎の"time complexity"がかなり早いということである.

References

[1] Jakub Konecny, H Brendan McMahan, Felix X Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[2] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In NIPS, pages 118 - 128, 2017.

[3]] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. arXiv preprint arXiv:1705.05491, 2017.

[4] Jiashi Feng, Huan Xu, and Shie Mannor. Distributed robust learning. arXiv preprint arXiv:1409.5937, 2014.

[5] Lili Su and Nitin H Vaidya. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In PODC, pages 425 - 434. ACM, 2016.

[6] Lili Su and Nitin H Vaidya. Defending non-bayesian learning against adversarial attacks. ISDC, 2016.

[7] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized Byzantine-tolerant SGD. arXiv preprint arXiv:1802.10116, 2018.

[8] Dong Yin, Yudong Chen, Kanna Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. arXiv preprint arXiv:1803.01498, 2018.