Brief Announcement: Byzantine-Tolerant Machine Learning

Peva Blanchard
Swiss Federal Institute of Technology, Lausanne
peva.blanchard@epfl.ch

El Mahdi El Mhamdi*†
Swiss Federal Institute of Technology, Lausanne
elmahdi.elmhamdi@epfl.ch

Rachid Guerraoui
Swiss Federal Institute of Technology, Lausanne
rachid.guerraoui@epfl.ch

Julien Stainer‡
Swiss Federal Institute of Technology, Lausanne
julien.stainer@epfl.ch

ABSTRACT

We report on *Krum*, the first *provably* Byzantine-tolerant aggregation rule for distributed Stochastic Gradient Descent (SGD). Krum guarantees the convergence of SGD even in a distributed setting where (asymptotically) up to half of the workers can be malicious adversaries trying to attack the learning system.

CCS CONCEPTS

•Mathematics of computing → Stochastic processes; *Differential calculus;* **•Hardware → Robustness; •Theory of computation → Machine learning theory;**

KEYWORDS

Distributed Stochastic Gradient Descent ; Adversarial Machine Learning

1 INTRODUCTION

The increasing amount of data involved as well as the growing complexity of models has led to learning schemes that require a lot of computational resources. As a consequence, most industry-grade machine-learning implementations are now distributed [1]. For example, as of 2012, Google reportedly used 16.000 processors to train an image classifier [8]. However, distributing a computation over several machines induces a higher risk of failures, including crashes and computation errors. In the worst case, the system may undergo *Byzantine* failures [5], i.e., completely arbitrary behaviors of some of the machines involved. In practice, such failures may be due to stalled processes, or biases in the way the data samples are distributed among the processes.

A classical approach to mask failures in distributed systems is to use a state machine replication protocol [11], which requires however state transitions to be applied by all processes. In the case of distributed machine learning, this constraint can be seen in two ways: either *(a)* the processes agree on a sample of data based on which they update their local parameter vectors, or *(b)* they agree on how the parameter vector should be updated. In case *(a)*, the sample of data has to be transmitted to each process, which then has to perform a heavyweight computation to update its local parameter vector. This entails communication and computational costs that defeat the entire purpose of distributing the work. In case *(b)*, the processes have no way to check if the chosen update for the parameter vector has indeed been computed correctly on real data (a Byzantine process could have proposed the update). Byzantine failures may easily prevent the convergence of the learning algorithm. Neither of these solutions is satisfactory in a realistic distributed machine learning setting.

In fact, most learning algorithms today rely on a core component, namely *stochastic gradient descent* (SGD) [4]. In a machine learning setting, a cost function – depending on the *parameter vector* – is minimized based on stochastic estimates of its gradient. Distributed implementations of SGD [12] typically take the following form: a single parameter server is in charge of updating the parameter vector, while worker processes perform the actual update estimation, based on the share of data they draw from un unknown distribution. The parameter server executes synchronous rounds, during each of which, the parameter vector is broadcast to the workers. In turn, each worker computes an estimate of the update to apply (an estimate of the *gradient*), and the parameter server aggregates their results to finally update the parameter vector. Today, this aggregation is typically implemented through averaging [10], or variants of it [6,12].

The motivation of this work is the question of how a distributed SGD can be devised to tolerate f Byzantine processes among the n workers. We provide the first provable answer to this question.

Contributions. We first show that no linear combination (current approaches) of the updates proposed by the workers can tolerate a *single* Byzantine worker. A non-linear, *distance-based* choice function, that chooses, among the proposed vectors, the vector "closest to everyone else" (for example by taking the vector that minimizes the sum of the distances to every other vector), might look appealing. Yet, such a distance-based choice tolerates only a single Byzantine worker. Two Byzantine workers can collude, one helping the other to be selected, by moving the barycenter of all the vectors farther from the "correct area". We formulate a Byzantine resilience property capturing sufficient conditions for

*E.M. El Mhamdi's work is funded by the Swiss National Science Foundation under the grant 200021_169588 TARBDA (*a Theoretical Approach to Robustness in Biological Distributed Algorithms*).

†Corresponding author.

‡P. Blanchard and J. Stainer are supported in part by the European ERC Grant 339539

PODC'17, July 25–27, 2017, Washington, DC, USA.

DOI: http://dx.doi.org/10.1145/3087801.3087861

the parameter server's choice to tolerate f Byzantine workers. Essentially, to guarantee that the cost will decrease despite Byzantine workers, we require the parameter server's choice *(a)* to point, on average, to the same direction as the gradient and *(b)* to have statistical moments (up to the fourth moment) bounded above by a homogeneous polynomial in the moments of a correct estimator of the gradient. One way to ensure such a resilience property is to consider a *majority-based* approach, looking at every subset of $n - f$ vectors, and considering the subset with the smallest diameter. While this approach is more robust to Byzantine workers that propose vectors far from the correct area, its exponential computational cost is prohibitive. Interestingly, combining the intuitions of the *majority-based* and *distance-based* methods, we can choose the vector that is somehow the closest to its $n - f$ neighbors. Namely, the one that minimizes a distance-based criteria, but only within its $n - f$ neighbors. This is the main idea behind our choice function we call *Krum*[1]. We show (using techniques from multi-dimensional stochastic calculus) that our Krum function satisfies the resilience property aforementioned and the corresponding machine learning scheme converges. An important advantage of the Krum function is that it requires $O(n^2 \cdot d)$ local computation time, where d is the dimension of the parameter vector. This contrasts with the prohibitive $O(n^d)$ cost of approximate agreement [9]. (In deep learning, the dimension d of the parameter vector may take values in the hundreds of billions.)

2 MODEL

We consider a general distributed system consisting of a parameter server[2] [1], and n workers, f of them possibly Byzantine. Computation is divided into (infinitely many) synchronous rounds. During round t, the parameter server broadcasts its parameter vector $x_t \in \mathbb{R}^d$ to all the workers. Each correct worker p computes an estimate $V_p^t = G(x_t, \xi_p^t)$ of the gradient $\nabla Q(x_t)$ of the cost function Q, where ξ_p^t is a random variable representing, e.g., the sample drawn from the dataset. A Byzantine worker b proposes a vector V_b^t which can be arbitrary (see Figure 1).

The parameter server computes a vector $F(V_1^t, \ldots, V_n^t)$ by applying a deterministic function F to the vectors received. We refer to F as the *choice function* of the parameter server. The parameter server updates the parameter vector using the following SGD equation:$x_{t+1} = x_t - \gamma_t \cdot F(V_1^t, \ldots, V_n^t)$. We assume that the correct (non-Byzantine) workers compute unbiased estimates of the gradient $\nabla Q(x_t)$. More precisely, in every round t, the vectors V_i^t's proposed by the correct workers are independent identically distributed random vectors, $V_i^t \sim G(x_t, \xi_i^t)$ with $\mathbb{E}G(x_t, \xi_i^t) = \nabla Q(x_t)$. This can be achieved by ensuring that each sample of data used for computing the gradient is drawn uniformly and independently, as classically assumed in the literature of machine learning [3]. The Byzantine workers have full knowledge of the system, including the choice function F, the vectors proposed by the other workers and can collaborate with each other [7].

[1]Krum, in Greek Κρούμος, was a Bulgarian Khan of the end of the eighth century, who undertook offensive attacks against the Byzantine empire. Bulgaria doubled in size during his reign.

[2]The parameter server is assumed to be reliable. Classical techniques of state-machine replication can be used to avoid this single point of failure.

3 BYZANTINE RESILIENCE

In most SGD-based learning algorithms used today [3,4], the choice function consists in computing the average of the input vectors. Lemma 3.1 below states that no linear combination of the vectors can tolerate a single Byzantine worker. In particular, averaging is not robust to Byzantine failures.

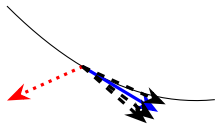

Figure 1: The gradient estimates of correct workers (black dashed arrows) are distributed around the actual gradient (blue solid arrow) of the cost function (thin black curve). A Byzantine worker can propose an arbitrary vector (red dotted arrow).

LEMMA 3.1. *Consider a choice function F_{lin} of the form: $F_{lin}(V_1, \ldots, V_n) = \sum_{i=1}^n \lambda_i \cdot V_i$, where the λ_i's are non-zero scalars. Let U be any vector in $\mathbb{R}^d$. A single Byzantine worker can make F always select U. In particular, a single Byzantine worker can prevent convergence.*

Intuitively, the choice function should output a vector F that is not too far from the "real" gradient g, more precisely, the vector that points to the steepest direction of the cost function being optimized. This is expressed as a lower bound (condition *(i)*) on the scalar product of the (expected) vector F and g. If $\mathbb{E}F$ belongs to the ball centered at g with radius r, then the scalar product is bounded below by a term involving $\sin\alpha = r/\|g\|$. Condition *(ii)* is more technical, and states that the moments of F should be controlled by the moments of the (correct) gradient estimator G. The bounds on the moments of G are classically used to control the effects of the discrete nature of the SGD dynamics [3]. Condition *(ii)* allows to transfer this control to the choice function.

Definition 3.2 ((α, f)-Byzantine Resilience). Let $0 \leq \alpha < \pi/2$ be any angular value, and any integer $0 \leq f \leq n$. Let $V_1, \ldots, V_n$ be any independent identically distributed random vectors in $\mathbb{R}^d$, $V_i \sim G$, with $\mathbb{E}G = g$. Let $B_1, \ldots, B_f$ be any random vectors in $\mathbb{R}^d$, possibly dependent on the V_i's. Choice function F is said to be (α, f)-Byzantine resilient if, for any $1 \leq j_1 < \cdots < j_f \leq n$, the vector $F = F(V_1, \ldots, \underbrace{B_1}_{j_1}, \ldots, \underbrace{B_f}_{j_f}, \ldots, V_n)$ satisfies *(i)* $\langle \mathbb{E}F, g \rangle \geq (1 - \sin\alpha) \cdot \|g\|^2 > 0$ and *(ii)* for $r = 2, 3, 4$, $\mathbb{E}\|F\|^r$ is bounded above by a linear combination of terms $\mathbb{E}\|G\|^{r_1} \ldots \mathbb{E}\|G\|^{r_{n-1}}$ with $r_1 + \cdots + r_{n-1} = r$.

4 THE KRUM FUNCTION

The barycentric choice function $F_{bary} = \frac{1}{n}\sum_{i=1}^n V_i$ can be defined as the vector in $\mathbb{R}^d$ that minimizes the sum of squared distances to the V_i's $\sum_{i=1}^n \left\|F_{bary} - V_i\right\|^2$. Lemma 3.1, however, states that this approach does not tolerate even a single Byzantine failure. One could try to define the choice function in order to select, *among* the V_i's, the vector $U \in \{V_1, \ldots, V_n\}$ that minimizes the sum $\sum_i \|U - V_i\|^2$. Intuitively, vector U would be close to every proposed vector, including the correct ones, and thus would be

close to the "real" gradient. However, all Byzantine workers but one may propose vectors that are large enough to move the total barycenter far away from the correct vectors, while the remaining Byzantine worker proposes this barycenter. Since the barycenter always minimizes the sum of squared distance, this last Byzantine worker is certain to have its vector chosen by the parameter server. This situation is depicted in Figure 2. In other words, since this choice function takes into account all the vectors, including the very remote ones, the Byzantine workers can collude to force the choice of the parameter server.

Figure 2: Selecting the vector that minimizes the sum of the squared distances to other vectors does not prevent arbitrary vectors proposed by Byzantine workers from being selected if $f \geq 2$. If the gradients computed by the correct workers lie in area C, the Byzantine workers can collude to propose up to $f-1$ vectors in an arbitrarily remote area B, thus allowing another Byzantine vector b, close to the barycenter of proposed vectors, to be selected.

Our approach to circumvent this issue is to preclude the vectors that are too far away. More precisely, we define our *Krum* choice function $\text{KR}(V_1, \ldots, V_n)$ as follows. For any $i \neq j$, we denote by $i \rightarrow j$ the fact that V_j belongs to the $n-f-2$ closest vectors to V_i. Then, we define for each worker i, the *score* $s(i) = \sum_{i \rightarrow j} \left\|V_i - V_j\right\|^2$ where the sum runs over the $n-f-2$ closest vectors to V_i. Finally, $\text{KR}(V_1, \ldots, V_n) = V_{i_*}$ where i_* refers to the worker minimizing the score, $s(i_*) \leq s(i)$ for all i.[3]

Lemma 4.1. *The Krum Function $\text{KR}(V_1, \ldots, V_n)$, where $V_1, \ldots, V_n$ are d-dimensional vectors, is computed in $O(n^2 \cdot d)$ time at the parameter server.*

Resilience. Proposition 4.2 below states that, if $2f+2 < n$ and the gradient estimator is accurate enough, (its standard deviation is relatively small compared to the norm of the gradient), then the Krum function is (α, f)-Byzantine-resilient, where angle α depends on the ratio of the deviation over the gradient. When the Krum function selects a correct vector (i.e., a vector proposed by a correct worker), the proof of this fact is relatively easy, since the probability distribution of this correct vector is that of the gradient estimator G. The core difficulty occurs when the Krum function selects a Byzantine vector (i.e., a vector proposed by a Byzantine worker), because the distribution of this vector is completely arbitrary, and may even depend on the correct vectors. In a very general sense, this part of our proof is reminiscent of the geometric median technique and is discussed in details in the full paper [2].

Proposition 4.2. *Let $V_1, \ldots, V_n$ be any independent and identically distributed random d-dimensional vectors s.t $V_i \sim G$, with $\mathbb{E}G = g$ and $\mathbb{E}\left\|G-g\right\|^2 = d\sigma^2$. Let $B_1, \ldots, B_f$ be any f random vectors, possibly dependent on the V_i's. If $2f+2 < n$ and $\eta(n,f)\sqrt{d}\cdot\sigma < \|g\|$, where $\eta(n,f) = \begin{cases} O(n) & \text{if } f = O(n) \\ O(\sqrt{n}) & \text{if } f = O(1) \end{cases}$, then the Krum function KR is (α, f)-Byzantine resilient where $0 \leq \alpha < \pi/2$ is defined by $\sin\alpha = \frac{\eta(n,f)\cdot\sqrt{d}\cdot\sigma}{\|g\|}$.*

Convergence. The SGD equation is expressed as follows: $x_{t+1} = x_t - \gamma_t \cdot \text{KR}(V_1^t, \ldots, V_n^t)$, where at least $n-f$ vectors among the V_i^t's are correct, while the other ones may be Byzantine. For a correct index i, $V_i^t = G(x_t, \xi_i^t)$ where G is the gradient estimator. We define the *local standard deviation* $\sigma(x)$ by $d \cdot \sigma^2(x) = \mathbb{E}\left\|G(x,\xi) - \nabla Q(x)\right\|^2$.

Proposition 4.3. *We assume that* (i) *the cost function Q is three times differentiable with continuous derivatives, and is non-negative, $Q(x) \geq 0$;* (ii) *the learning rates satisfy $\sum_t \gamma_t = \infty$ and $\sum_t \gamma_t^2 < \infty$;* (iii) *the gradient estimator satisfies $\mathbb{E}G(x,\xi) = \nabla Q(x)$ and $\forall r \in \{2, \ldots, 4\}$, $\mathbb{E}\|G(x,\xi)\|^r \leq A_r + B_r\|x\|^r$ for some constants A_r, B_r;* (iv) *there exists a constant $0 \leq \alpha < \pi/2$ such that for all x $\eta(n,f)\cdot\sqrt{d}\cdot\sigma(x) \leq \|\nabla Q(x)\|\cdot\sin\alpha$;* (v) *finally, beyond a certain horizon, $\|x\|^2 \geq D$, there exist $\epsilon > 0$ and $0 \leq \beta < \pi/2 - \alpha$ such that $\|\nabla Q(x)\| \geq \epsilon > 0$ and $\frac{\langle x, \nabla Q(x)\rangle}{\|x\|\cdot\|\nabla Q(x)\|} \geq \cos\beta$. Then the sequence of gradients $\nabla Q(x_t)$ converges almost surely to zero.*

Proposition 4.3 basically says that in the presence of Byzantine workers, the parameter vector x_t almost surely reaches a basin around points where the gradient is small ($\|\nabla Q\| \leq \eta(n,f)\cdot\sqrt{d}\cdot\sigma$), i.e., points where the cost landscape is "almost flat".

Note that the convergence analysis is based only on the fact that function KR is (α, f)-Byzantine resilient. The detailed proof of convergence, as well as its qualitative interpretation, can be found in the full paper [2].

Acknowledgment. We are very grateful to Lê Nguyen Hoang for fruitful discussions.

[3] If two or more workers have the minimal score, we choose the vector of the worker with the smallest identifier.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Byzantine-tolerant machine learning. *arXiv preprint arXiv:1703.02757*, 2017.

[3] L. Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, 17(9):142, 1998.

[4] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[6] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

[7] N. A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[8] J. Markoff. How many computers to identify a cat? 16,000. *New York Times*, pages 06–25, 2012.

[9] H. Mendes and M. Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 391–400. ACM, 2013.

[10] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[11] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[12] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.