Fitting ReLUs via SGD and Quantized SGD

Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and A. Salman Avestimehr
Ming Hsieh Department of Electrical Engineering, University of Southern California
Email: mmousavi@usc.edu, soltanol@usc.edu, avestimehr@ee.usc.edu

Abstract

In this paper we focus on the problem of finding the optimal weights of the shallowest of neural networks consisting of a single Rectified Linear Unit (ReLU). These functions are of the form $\boldsymbol{x} \to \max(0, \langle \boldsymbol{w}, \boldsymbol{x} \rangle)$ with $\boldsymbol{w} \in \mathbb{R}^d$ denoting the weight vector. We focus on a planted model where the inputs are chosen i.i.d. from a Gaussian distribution and the labels are generated according to a planted weight vector. We first show that mini-batch stochastic gradient descent when suitably initialized, converges at a geometric rate to the planted model with a number of samples that is optimal up to numerical constants. Next we focus on a parallel implementation where in each iteration the mini-batch gradient is calculated in a distributed manner across multiple processors and then broadcast to a master or all other processors. To reduce the communication cost in this setting we utilize a Quanitzed Stochastic Gradient Scheme (QSGD) where the partial gradients are quantized. Perhaps unexpectedly, we show that QSGD maintains the fast convergence of SGD to a globally optimal model while significantly reducing the communication cost. We further corroborate our numerical findings via various experiments including distributed implementations over Amazon EC2.

1 Introduction

Many modern learning tasks involve fitting nonlinear models to data. Given training data consisting of n pairs of input features $\boldsymbol{x}_i \in \mathbb{R}^d$ and desired outputs $y_i \in \mathbb{R}$ we wish to infer a function that best explains the training data. A prominent example is neural network models which have enabled impressive empirical success in applications spanning natural language processing to robotics. Guaranteed training of nonlinear data models however remain elusive. The main challenge is that fitting such nonlinear models requires solving highly nonconvex optimization problems and it is not clear why local search methods such as stochastic gradient descent converge to globally optimal solutions without getting stuck in spurious local optima and saddles.

In this paper we focus on fitting Rectified Linear Units (ReLUs) to the data which are functions $\phi_{\boldsymbol{w}} : \mathbb{R}^d \to \mathbb{R}$ of the form $\phi_{\boldsymbol{w}}(\boldsymbol{x}) = \max(0, \langle \boldsymbol{w}, \boldsymbol{x} \rangle)$. We study a nonlinear least-squares formulation of the form

$$\begin{aligned} \min_{\boldsymbol{w} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{w}) :=& \frac{1}{n} \sum_{i=1}^{n} \ell_i(\boldsymbol{w}) \\ =& \frac{1}{n} \sum_{i=1}^{n} \big(\max(0, \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) - y_i\big)^2. \end{aligned} \tag{1.1}$$

A popular approach to solving problems of this kind is via Stochastic Gradient Descent (SGD). Indeed, SGD due to its manageable memory and footprint and highly parallelizable nature has

become a mainstay of modern learning systems. Despite its wide use however, due to the nonconvex nature of the loss it is completely unclear why SGD converges to a globally optimal model. Fitting ReLUs via SGD poses new challenges: When are the iterates able to converge to global optima? How many samples are required? What is the convergence rate and is it possible to insure a fast, geometric rate of convergence? How do the answers to above change based on the mini-batch size?

Yet, another challenge that arises when implementing SGD in a distributed framework is the high communication overhead required for transferring gradient updates between processors. A recent remedy is the use of quantized gradient updates such as Quantized SGD (QSGD) to reduce communication overhead. However, there is little understanding of how such quantization schemes perform on nonlinear learning tasks. Do quantized updates converge to the same solution as unquantized variants? If so, how is the convergence rate affected? How many quantization levels or bits are required to achieve good accuracy?

In this paper we wish to address the above challenges. Our main contributions are as follows:

- We study the problem of fitting ReLUs and show that SGD converges at a fast linear rate to a globally optimal solution. This holds with a near minimal number of data observations. We also characterize the convergence rate as a function of the SGD mini-batch sizes.
- We show that the QSGD approach of [1] also converges at a linear rate to a globally optimal solutions. This holds even when the number of quantization levels grows only Logarithmically in problem dimension. We also characterize the various tradeoffs between communication and computational resources when using such low-precision algorithms.
- We provide experimental results corroborating our theoretical findings.

2 Algorithms: SGD and QSGD

In this section we discuss the details of the algorithms we plan to study. We begin by discussing the specifics of the SGD iterates we will use. We then discuss how to use quantization techniques in order to reduce the communication overhead in a distributed implementation.

2.1 Stochastic Gradient Descent (SGD) for fitting ReLUs

To solve the optimization problem (1.1) we use a mini-batch SGD scheme. While, the loss function (1.1) is not differentiable, one can still use an update akin to SGD by defining a generalized notion of gradients for non-differentiable points as a limit of gradients of points converging to the non-differentiable point [2]. Then, in each iteration we sample the indices $i_t^{(1)}, i_t^{(2)}, ..., i_t^{(m)}$ uniformly with replacement from $\{1, 2, \ldots, n\}$ and apply updates of the form

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \frac{1}{m} \sum_{j=1}^{m} \nabla \ell_{i_t^{(j)}}(\boldsymbol{w}_t). \tag{2.1}$$

Here, $\nabla \ell_i$ denotes the generalized gradient of the loss ℓ_i and is equal to

$$\nabla \ell_i(\boldsymbol{w}) = 2\big(\mathrm{ReLU}\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle\right) - y_i\big)\big(1 + \mathrm{sgn}(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)\big)\boldsymbol{x}_i.$$

2.2 Reducing the communication overhead via Quantized SGD

One of the major advantages of SGD is that it is highly scalable to massive amounts of data. SGD can be easily implemented in a distributed platform where each processor calculates a portion of the mini-batch based on the available local data. Then the partial gradients are sent back to a master node or the other processors to calculate the full mini-batch gradient and update the next iteration. The latter case for example is common in modern deep learning implementations [1]. Both distributed approaches however, suffer from a major bottleneck due to the cost of transmitting the gradients to the master or between the processors.

A recent remedy for reducing this cost is utilizing lossy compression to quantize the gradients prior to transmission/broadcast. In particular, a recent paper [1] proposes the quantized SGD (QSGD) algorithm based on a randomized quantization function $Q_s : \mathbb{R}^d \to \mathbb{R}^d$ with s denoting the number of quantization levels. Specifically, for a vector $\boldsymbol{v} \in \mathbb{R}^d$ the ith entry of the quantization function $Q_s(\boldsymbol{v})$ is given by

$$Q_s(v_i) = \|\boldsymbol{v}\|_2 \cdot \mathrm{sgn}(v_i) \cdot \xi_i(\boldsymbol{v}, s),$$

where $\xi_i(\boldsymbol{v}, s)$'s are independent random variables defined as

$$\xi_i(\boldsymbol{v}, s) = \begin{cases} h/s & \text{with prob. } 1 - (\frac{|v_i|s}{\|\boldsymbol{v}\|_2} - h). \\ (h+1)/s & \text{otherwise.} \end{cases}$$

Here, $h \in [0, s)$ is an integer such that $\frac{|v_i|}{\|\boldsymbol{v}\|_2} \in [h/s, (h+1)/s]$ and we follow the convention that $\mathrm{sign}(0) = 0$.

To see how this quantization scheme can be used in a distributed setting consider a master-worker distributed platform consisting of K worker processors numbered $1, 2, ..., K$ and a master processor. To run SGD on this platform we partition the n training data points into K batches of size $\frac{n}{K}$ with each worker processor storing one of the batches. In each iteration, the master broadcasts the latest model to all the workers. Each worker then chooses m_k points from local available data points randomly and computes a partial gradient based on the selected data points using the latest model received from the master and quantizes the resulting partial gradient. The workers then send the quantized stochastic gradients to the master. The master waits for all the quantized partial stochastic gradients from the workers and then updates the model using their average. As a result the aggregate effect of QSGD leads to updates of the form

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \frac{1}{K} \sum_{k=1}^{K} Q_s \left(\nabla \left\{ \frac{1}{m_k} \sum_{j=1}^{m_k} \ell_{i_t^{(j)}}(\boldsymbol{w}_t) \right\} \right). \tag{2.2}$$

Since only the quantized partial gradients are transimitted between the processors, QSGD significantly reduces the number of communicated bits.

3 Main results

3.1 SGD for fitting ReLUs

In this section we discuss our results for convergence of the SGD iterates.

Theorem 3.1 *Let $\boldsymbol{w}^*$ be a fixed weight vector, and the feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ be i.i.d. Gaussian random vectors distributed as $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with the corresponding labels given by $y_i = max(0, \langle x_i, \boldsymbol{w}^* \rangle)$. Furthermore, assume*

(I) the number of samples obey $n > c_0 d$, for a fixed numerical constant c_0.

(II) the initial estimate $\boldsymbol{w}_0$ obeys $||\boldsymbol{w_0} - \boldsymbol{w}^*||_2 \leq \sqrt{\delta_1}\frac{7}{200}||\boldsymbol{w}^*||_2$ for some $0 < \delta_1 \leq 1/2$.

Then, the Stochastic Gradient Descent (SGD) updates in (2.1) with mini-batch size $m \in \mathbb{N}$ and learning rate $\eta = \frac{3}{4(\frac{9d}{m} + \frac{25}{16})}$ obey

$$\mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2] \leq \underbrace{\left(1 - \frac{9}{16(\frac{9d}{m} + \frac{25}{16})}\right)}_{\text{convergence rate}=\rho}^t \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 \tag{3.1}$$

with probability at least $1 - \delta_1 - 2(n+2)e^{-\gamma d} - 2(n+25)e^{-\gamma n}$. Furthermore, if $t \geq (\log(2/\epsilon) + \log(1/\delta_2))\frac{1}{1-\rho}$ for $0 < \delta_2 \leq 1$, then

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2 \leq \epsilon \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2, \tag{3.2}$$

holds with probability at least $1 - \delta_1 - \delta_2 - 2(n+2)e^{-\gamma d} - 2(n+25)e^{-\gamma n}$.

Remark 3.2 *We note that Theorem 3.1 requires the initial estimate $\boldsymbol{w}_0$ to be sufficiently close to the planted model (i.e. $\leq \sqrt{\delta_1}\frac{7}{200}||\boldsymbol{w}^*||_2$). Such an initial estimate can be easy obtained by using one full gradient descent step at zero [3]. The required sample complexity for this initialization to be effective is on the order of $n \geq c\frac{d}{\delta_1^2}$ for c a fixed numerical constant. For the purposes of this result we will use δ_1 a small constant so that on the order of $n \gtrsim d$ samples are sufficient for this initialization.*

Remark 3.3 *Theorem 3.1 shows that the SGD iterates (2.1) converge to a globally optimal solution at a geometric rate. Furthermore, the required number of samples for this convergence to occur is nearly minimal and is on the order of the number of parameters d.*

Remark 3.4 *Theorem 3.1 also characterizes the influence of mini-batch size on the convergence rate, illustrating the trade-off between the computational load and the convergence speed.*

Remark 3.5 *In Theorem 3.1, the probability of success depends on the distance of initial point to the optimal solution. As we detail in the appendix we can use an ensemble algorithm to reduce the failure probability arbitrarily small without the need to start from an initial point that is very close to the optimal solution.*

Remark 3.6 *We note that for $m = n$, the updates in (2.1) reduce to full gradient descent and the guarantee (3.1) takes the form $||\boldsymbol{w}_t - \boldsymbol{w}^*||_2^2 \leq \rho^t ||\boldsymbol{w}_0 - \boldsymbol{w}^*||_2^2$. In this special case our result recovers that of [3] up to a constant factor.*

3.7 Quantized SGD

We next focus on providing gurantees for QSGD.

Theorem 3.2 *Consider the same setting and assumptions as Theorem 3.1. Furthermore, consider a parallel setting with K worker processors and a master per Section 2.2 and assume that each worker computes $\frac{m}{K}$ partial gradients in each iteration (i.e. $m_k = m/K$). We run QSGD over these processors via the iterative updates in (2.2). Then*

$$\mathbb{E}[\|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2] \leq \underbrace{\left(1 - \frac{9}{16\left(\left(1 + min(\frac{d}{s^2}, \frac{\sqrt{d}}{s})\right)(\frac{9d}{m} + \frac{25}{16}) + \frac{25}{16}\right)}\right)^t}_{convergence\ rate=\alpha} \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 \tag{3.3}$$

holds with probability at least $1-\delta_1-2(n+2)e^{-\gamma d}-2(n+25)e^{-\gamma n}$. Furthermore, if $t \geq (\log(2/\epsilon) + \log(1/\delta_2))\frac{1}{1-\alpha}$ for $0 < \delta_2 \leq 1$ then long

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2 \leq \epsilon \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2 \tag{3.4}$$

with probability at least $1 - \delta_1 - \delta_2 - 2(n+2)e^{-\gamma d} - 2(n+25)e^{-\gamma n}$.

Remark 3.8 *As mentioned in Remark 3.2 of Theorem 3.1, in order to address the initialization issue we can use one full Gradient Descent pass from zero to find an initializaiton obeying the conditions of this theorem.*

Remark 3.9 *Similar to the results of Theorem 3.1, Theorem 3.2 shows geometric convergence of the QSGD iterates (2.2) with a near minimal number of samples ($n \gtrsim d$). Furthermore, it characterizes the effect of both mini-batch size and quantization levels on the convergence rate. Specifically, by increasing the quantization levels, the iterates (2.1) converge faster. Perhaps unexpected, by choosing the number of the bits to be on the order of $\log\sqrt{d}$ the iterates (2.1) and (2.2) converge with the same rate up to a constant factor. This allows QSGD to significantly reduce the communication load while maintaining a computational effort comparable to SGD.*

4 Numerical results and experiments on Amazon EC2

In this section we wish to investigate the results of Theorems 3.1 and 3.2 using numerical simulations and experiments on Amazon EC2. We first wish to investigate how the rate of convergence of mini-batch SGD and QSGD iterates depends on the different parameters. To this aim, we generate the planted weight vector $\boldsymbol{w}^* \in \mathbb{R}^d$ with $d = 1000$ with entries distributed i.i.d. $\sim N(200, 3)$. In addition, we generate $n = 10000$ feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ i.i.d. $\sim N(0, 1)$ and set the corresponding output labels $y_i = \max(0, \langle \boldsymbol{x}_i, \boldsymbol{w}^* \rangle)$. To estimate $\boldsymbol{w}^*$, we start from a random initial point and run SGD and QSGD with learning rates $\eta = \frac{m}{d}$ and $\frac{m \cdot b}{9d}$, where b is the number of bits required for quantization.

In Figure 1(a) we focus on corroborating our convergence analysis for SGD. To this aim we vary the mini-batch size m and plot the relative error ($\|\boldsymbol{w} - \boldsymbol{w}^*\|_2 / \|\boldsymbol{w}^*\|_2$) as a function of the iterations.

This figure demonstates that the convergence rate is indeed linear and increasing the batch size m results in a faster convergence.

In Figure 1(b) we focus on understanding the effect of the number of bits on the convergence behavior of QSGD. To this aim we fix the mini-batch size at $m = 800$ and vary the number of bits b. This plot confirms that QSGD maintains a linear convergence rate comparable to SGD (especially when $b = 7$).

To understand the required sample complexity of the algorithms, we plot phase transition curves depicting the empirical probability that gradient descent converges to $\boldsymbol{w}^*$ for different data set sizes (n) and feature dimensions (d). For each value of n and d we perform 10 trials with the data generated according to the model discussed above. In each trial we run the algorithm for 2000 iterations. If the relative error after 2000 iterations is less than 10^{-3} we consider the trial a success otherwise a failure. Figures 2(a) and 2(b) depict the phase transition for mini-batch SGD and QSGD with $b = 7$ bits, respectively. As it can be seen, the figures corroborate the linear relationship of the required sample complexity with the feature dimensions. Furthermore, these figures demonstrate that quantization does not substantially change the required sample complexity.

In order to understand the effectiveness of QSGD at reducing the communication overhead, we provide experiments on Amazon EC2 clusters and compare the performance of QSGD with SGD. We utilize a master-worker architecture and make use of **t2.micro** instances. We use Python for implementing two algorithms and use MPI4py [4] for message passing across the instances. Before starting the iterative updates each worker receives its portion of the training data. In each iteration t, after the workers receive the latest model $\boldsymbol{w}_t$ from the master, they compute the stochastic gradients at $\boldsymbol{w}_t$ based on the local data and then send it back to the master. In SGD and QSGD we use *float64* and *int8* for sending gradients, respectively. Additionally, we use `isend()` and `irecv()` for sending and receiving, respectively, and `Time.time()` to measure the running time.

We compare the performances of these two algorithms in the following two scenarios.

- Scenario one: We use 41 **t2.micro** instances, with one master and 40 workers. We have $n = 20000$ data points with feature dimension $d = 4000$. We partition and distribute the data into 40 equal batches of size $20000/40 = 500$ with each worker performing updates based its own batch.

- Scenario two: We use 51 **t2.micro** instances, with one master and 50 workers. In this scenario we use $n = 25000$ and $d = 4000$, and again distribute the data evenly among the workers.

Table 1 summarizes the experiment scenarios. In both cases we run SGD and QSGD algorithms for 300 iterations. Figure 3 depicts the total running times. Table 2 also shows the breakdowns of the run-times. These experiments indicate that the total running time of QSGD is 5 times less than that of SGD. Since both algorithms have similar convergence rate and hence computational time, this clearly demonstrates that the communication time for QSGD is significantly smaller.

Table 1: Experiment scenarios.

scenario index	# of workers (K)	# of data points (n)	feature dimension (d)
1	40	20000	4000
2	50	25000	4000

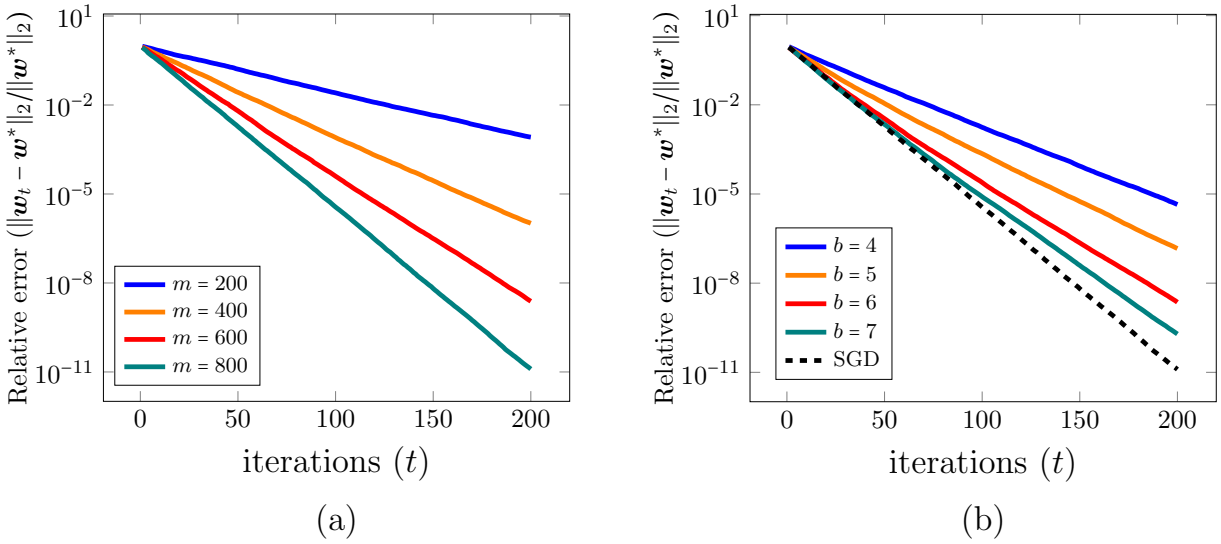
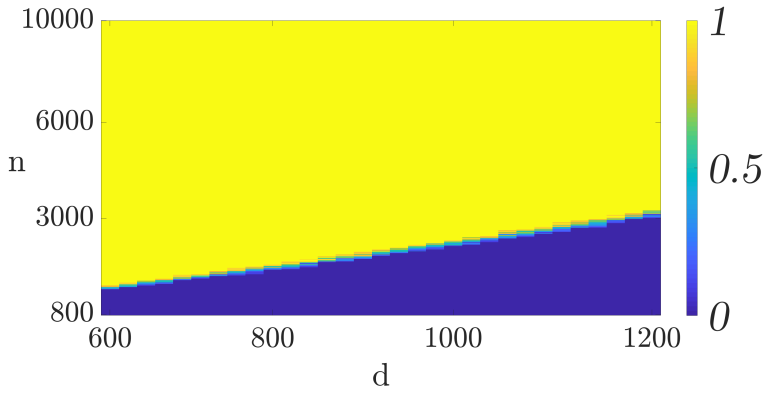


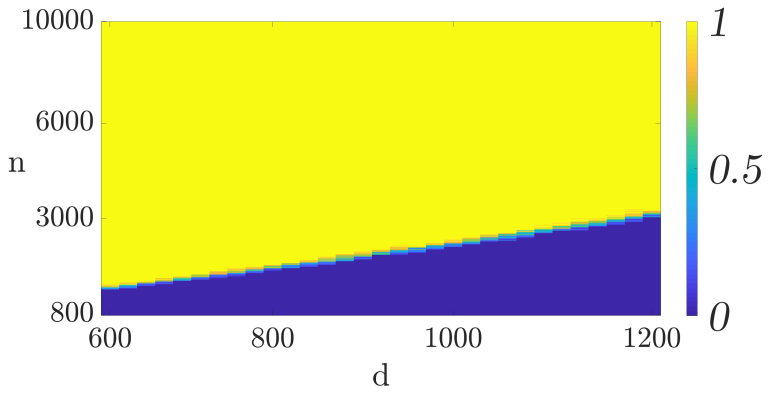
Figure 1: (a) This plot depicts the convergence behavior of mini-batch SGD iterates for various mini-batch sizes m.(b) This plot depicts the convergence behavior of QSGD iterates for various bits of quantization b with the mini-batch size fixed at $m = 800$.

Table 2: Breakdowns of the run-times in the both scenarios.

schemes	scenario index	comm. time	comp. time	total time
SGD	1	28.5100 s	4.921 s	33.431 s
QSGD	1	3.2470 s	5.056 s	8.303 s
SGD	2	38.2910 s	4.94 s	43.231 s
QSGD	2	6.5010 s	5.169 s	11.67 s



(a)



(b)

Figure 2: Empirical probability that (a) SGD and (b) QSGD with $b = 7$ finds the global optimum for different number of data points (n) and feature dimensions (d).

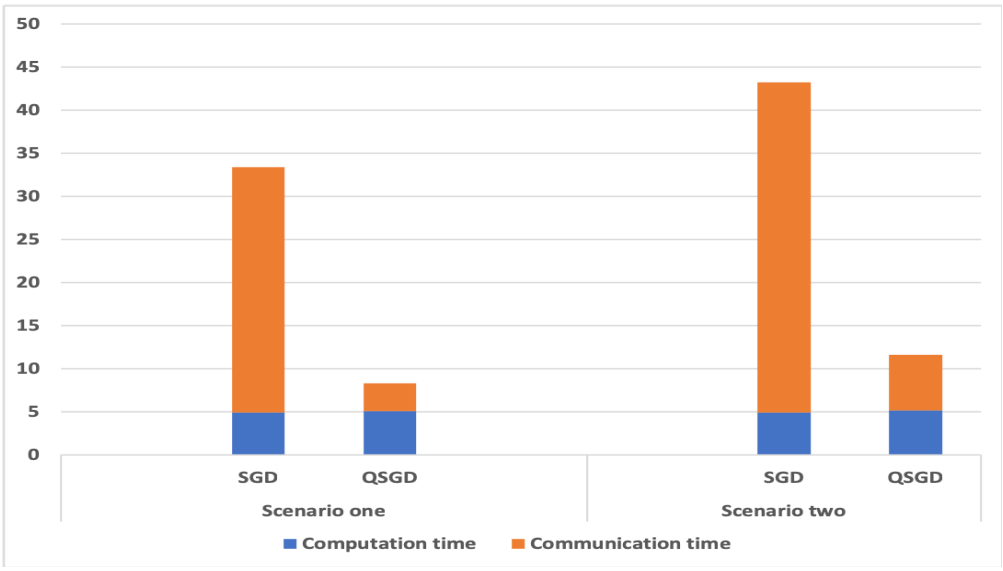


Figure 3: Run-time compassion of SGD and QSGD on Amazon EC2 clusters for the two scenarios.

5 Related work

The problem of fitting non-linear models to data has a rich history in statistics and learning theory using a variety of algorithms [5, 6, 7, 8]. In the context of training deep models such nonlinear learning problems have lead to major breakthroughs in various applications [9, 10]. Despite these theoretical and empirical advances, rigorous understanding of local search heuristics such as stochastic gradient descent which are arguably the most widely used techniques in practice, have remained elusive.

Recently, there has been a surge of activity surrounding nonlinear data fitting via local search. Focusing on ReLU nonlinearities [3] shows that in generic instances full gradient descent converges to a globally optimal model at a geometric rate. See also [11, 12, 13, 14, 15, 16, 17, 18] for a variety of related theoretical and empirical work for fitting ReLUs and shallow neural networks via local search heuristics. In contrast to the above, which require full gradient updates in this paper we focus on fitting ReLU nonlinearities via mini-batch SGD and QSGD. Finally, we would like to mention another recent result which studies a stochastic method for fitting ReLUs [19] via a First order Perturbed Stochastic Gradient Descent. This approach differs from ours in a variety of ways including the update strategy and required sample complexity. In particular, this approach is based on stochastic methods with random search strategies which only use function evaluations. Furthermore, this result requires $n \gtrsim d^4$ samples and provides a polynomial converge guarantee where as is this paper we have established a geometric rate of convergence to the global optima with a near minimal number of samples that scales linearly in the problem dimension (i.e. $n \gtrsim d$).

Recently there has been a lot of exciting activity surrounding SGD analysis. Classic SGD convergence analysis shows that the distance to the global optima (in loss or parameter value) decreases polynomially in the number of iterations (e.g. $1/t$ after t iterations). More recent results demonstrate that in certain cases, a significantly faster and geometric rate of convergence (i.e. ρ^{-t} with $\rho < 1$) is possible [20, 21]. For instance, [22] showed that randomized Kaczmarz algorithm

converges to the solution of a consistent linear system of equations at a geometric rate. More recently, [23] showed that under some assumptions such as smoothness, strong convexity, and perfect interpolation, SGD achieves a geometric rate of convergence. In this paper we add to this growing literature and demonstrate that such a fast geometric rate of convergence to a globally optimal solution is also possible despite the nonlinear and nonconvex nature of fitting ReLUs.

Reducing communication overhead by compressing the gradients has become increasing popular in recent literature [24, 25, 26]. Most notably [27] empirically demonstrated that one-bit quantization of gradients is highly effective for scalable training of deep models. The QSGD paper [1] develops convergence gurantees for convex losses as well convergence of gradient to zero for nonconvex losses. Related [28] also shows that under the assumption of smoothness and bounded variance a quantized SGD procedure converges to a stationary point of a general non-convex function with a polynomial convergence rate. In contrast in this paper we focus on geometric convergence to the global optima but for the specific problem of fitting a ReLU nonlinearity.

6 Proofs

6.1 Convergence analysis for fitting ReLUs via SGD (Proof of Theorem 3.1)

To show

$$\mathop{\mathbb{E}}_{\boldsymbol{w}_t}\left[\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2\right]\leq\rho\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2, \tag{6.1}$$

we begin with a useful definition. In particular, we denote the mini-batch empirical loss as $L_{I_m(t)}(\boldsymbol{w})\triangleq\frac{1}{m}\sum_{j=1}^m\ell_{i_t^{(j)}}(\boldsymbol{w})$, where $I_m(t)=\{i_t^{(1)},i_t^{(2)},...,i_t^{(m)}\}$ is the set of indices chosen at iteration t. Therefore, we can rewrite the mini-batch SGD updates as

$$\boldsymbol{w}_t-\boldsymbol{w}^*=\boldsymbol{w}_{t-1}-\boldsymbol{w}^*-\eta\nabla L_{I_m(t)}(\boldsymbol{w}_{t-1}).$$

To upper bound $\mathbb{E}\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2$ in terms of $\mathbb{E}\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2$, we expand $\mathbb{E}\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2$ in the form

$$\begin{aligned}\mathbb{E}_{I_m(t)}\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2=&\mathbb{E}_{I_m(t)}\left[\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2\right]\\&+\mathbb{E}_{I_m(t)}\left[-2\eta\langle\boldsymbol{w}_{t-1}-\boldsymbol{w}^*,\nabla L_{I_m(t)}(w_{t-1})\rangle+\eta^2\|\nabla L_{I_m(t)}(w_{t-1})\|_2^2\right]\end{aligned} \tag{6.2}$$

To draw the desired conclusion of the theorem (i.e. (6.1)), it suffices to upper bound the second expectation which consists of two terms . To upper bound the first term, We apply the expectation to the inner product to conclude that

$$\mathbb{E}_{I_m(t)}\left[\langle\boldsymbol{w}_{t-1}-\boldsymbol{w}^*,\nabla L_{I_m(t)}(w_{t-1})\rangle\right]=\langle\boldsymbol{w}_{t-1}-\boldsymbol{w}^*,\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\rangle \tag{6.3}$$

To lower bound the inner product we utilize the following lemma proved in [3].

Lemma 6.1 *For the loss function defined in (1.1), we have*

$$\langle\boldsymbol{u},\boldsymbol{w}-\boldsymbol{w}^*-\nabla\mathcal{L}(\boldsymbol{w})\rangle\ \leq 2\left(\delta+\sqrt{1+\delta}\left(1+\frac{2}{(1-\epsilon)^2}\right)\left(\delta+\sqrt{\frac{21}{20}}\epsilon\right)\right)\cdot\|\boldsymbol{w}-\boldsymbol{w}^*\|_2,$$

holding for all $u \in \mathcal{B}^d$ *and* $w \in E(\epsilon) = \left\{\boldsymbol{w} \in \mathbb{R}^d : \|\boldsymbol{w} - \boldsymbol{w}^*\|_2 \le \epsilon \|\boldsymbol{w}^*\|_2\right\}$ *with probability at least* $1 - 16e^{-\gamma\delta^2 n} - (n+10)e^{-\gamma n}$. *Specifically for* $\delta = 10^{-4}$ *and* $\epsilon = 7/200$, *we obtain*

$$\langle \boldsymbol{u}, \boldsymbol{w}_{t-1} - \boldsymbol{w}^* - \nabla \mathcal{L}(\boldsymbol{w}_{t-1}) \rangle \le \frac{1}{4} \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2. \tag{6.4}$$

Using Lemma 6.1 we have

$$\langle \boldsymbol{u}, \boldsymbol{w}_{t-1} - \boldsymbol{w}^* - \nabla \mathcal{L}(\boldsymbol{w}_{t-1}) \rangle \le \frac{1}{4} \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2$$

for all $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_2 \le 1$. By choosing $u = \frac{\boldsymbol{w}_{t-1} - \boldsymbol{w}^*}{\|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2}$ we can conclude that

$$\langle \boldsymbol{w}_{t-1} - \boldsymbol{w}^*, \nabla \mathcal{L}(w_{t-1}) \rangle \ge \frac{3}{4} \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2 \tag{6.5}$$

holds with probability at least $1 - 16e^{-10^{-8}\gamma n} - (n+10)e^{-\gamma n}$.

Thus, by combining (6.2), (6.3), and (6.5) we can conclude that

$$\mathbb{E}_{I_m(t)} \|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2 \le (1 - \frac{3\eta}{2}) \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2 + \eta^2 \mathbb{E}_{I_m(t)} \|\nabla L_{I_m(t)}(\boldsymbol{w}_{t-1})\|_2^2. \tag{6.6}$$

Next, in order to upper bound the second term in the right hand side of (6.6), we use following lemma proven in Section 6.1.1.

Lemma 6.2 *The following inequality*

$$\mathbb{E}_{I_m(t)} \|\nabla L_{I_m(t)}(\boldsymbol{w}_{t-1})\|_2^2 \le \left(\frac{9d}{m} + \frac{25}{16}\right) \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2 \tag{6.7}$$

holds with probability at least $1 - 2(n+1)e^{-\gamma d} - (n+25)e^{-\gamma n}$.

Hence, by (6.6) and (6.7) with $\eta^* = \frac{3}{4(\frac{9d}{m} + \frac{25}{16})}$ we arrive at

$$\begin{aligned}
\mathbb{E}_{I_m(t)} \|\boldsymbol{w}_t - \boldsymbol{w}^*\|_2^2 &\le \left[1 - \frac{3}{2}\eta + (\frac{9d}{m} + \frac{25}{16})\eta^2\right] \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2 \\
&\le (1 - \frac{9}{16(\frac{9d}{m} + \frac{25}{16})}) \mathbb{E} \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2 \\
&= \rho \cdot \mathbb{E} \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2^2.
\end{aligned} \tag{6.8}$$

This holds with probability at least $1 - 2(n+1)e^{-\gamma d} - 2(n+25)e^{-\gamma n}$. We note however, that the proof is not yet complete. We have not shown that all subsequent iterations will also lie in the local neighborhood dictated by Lemma 6.1. We have only shown that on *average* they belong to this neighborhood. To overcome this challenge we utilize some techniques from stochastic processes to obtain a conditional convergence. Our argument heavily borrows from [29] with some parts directly adapted.

Conditional linear convergence. In order to make our notations compatible with typical notations used in stochastic processes theory, we denote $\boldsymbol{W}_k$ as the random vector of estimated solution

at iteration k and $\boldsymbol{w}_k$ as a realization of that random vector. Using these conventions the result we have proven so far can be rewritten as

$$\mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2 | \boldsymbol{W}_k = \boldsymbol{w}_k] \leq \rho ||\boldsymbol{w}_k - \boldsymbol{x}||_2^2.$$

Let $\mathcal{F}_k$ denote the σ-algebra generated by indices chosen in the steps from 1 to k. Also let $B \subset R^d$ be the region consisting of all points which are in $E(\epsilon) = \left\{\boldsymbol{w} \in \mathbb{R}^d : ||\boldsymbol{w} - \boldsymbol{w}^*||_2 \leq \epsilon ||\boldsymbol{w}^*||_2\right\}$ where $\epsilon = \frac{7}{200}$. Finally, assume an initial estimate which is fixed obeying $\boldsymbol{w}_0 \in B$ and $||\boldsymbol{w}_0 - \boldsymbol{w}^*||_2 \leq \sqrt{\delta_1}\epsilon ||\boldsymbol{w}^*||_2$. Now define a stopping time τ as $\tau := \min\{k : \boldsymbol{W}_k \notin B\}$. For each k, and $\boldsymbol{w}_k \in B$, we have

$$\begin{aligned}
\mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2 1_{\tau > k+1} | \boldsymbol{W}_k = \boldsymbol{w}_k] \leq & \mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||^2 1_{\tau > k} | \boldsymbol{W}_k = \boldsymbol{w}_k] \\
= & \mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2 1_{\tau > k} | \boldsymbol{W}_k = \boldsymbol{w}_k, \mathcal{F}_k] \\
= & \mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2 | \boldsymbol{W}_k = \boldsymbol{w}_k, \mathcal{F}_k] 1_{\tau > k} \\
\leq & \rho ||\boldsymbol{w}_k - \boldsymbol{w}^*||_2^2 1_{\tau > k}
\end{aligned}$$

Hence

$$\mathbb{E}[||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2 1_{\tau > k+1}] \leq \rho \mathbb{E}[||\boldsymbol{W}_k - \boldsymbol{w}^*||_2^2 1_{\tau > k}]$$

Therefore, we obtain

$$\mathbb{E}[||\boldsymbol{W}_k - \boldsymbol{w}^*||_2^2 1_{\tau > k}] \leq \rho^k ||\boldsymbol{w}_0 - \boldsymbol{w}^*||_2^2. \tag{6.9}$$

To show that the probability of leaving this neighborhood is low we bound $\mathbb{P}(\tau < \infty)$. We begin by showing that $Z_k = \frac{||\boldsymbol{W}_{\tau \wedge k} - \boldsymbol{w}^*||_2^2}{\rho^{\tau \wedge k}}$ is a supermartingale.

$$\begin{aligned}
\mathbb{E}[Z_{k+1} | \mathcal{F}_k] = & \mathbb{E}[\frac{||\boldsymbol{W}_{\tau \wedge (k+1)} - \boldsymbol{w}^*||_2^2}{\rho^{\tau \wedge (k+1)}} 1_{\{\tau \leq k\}} | \mathcal{F}_k] + \mathbb{E}[\frac{||\boldsymbol{W}_{\tau \wedge (k+1)} - \boldsymbol{w}^*||_2^2}{\rho^{\tau \wedge (k+1)}} 1_{\{\tau > k\}} | \mathcal{F}_k] \\
= & \mathbb{E}[\frac{||\boldsymbol{W}_{\tau \wedge (k)} - \boldsymbol{w}^*||_2^2}{\rho^{\tau \wedge (k)}} 1_{\{\tau \leq k\}} | \mathcal{F}_k] + \mathbb{E}[\frac{||\boldsymbol{W}_{k+1} - \boldsymbol{w}^*||_2^2}{\rho^{k+1}} 1_{\{\tau > k\}} | \mathcal{F}_k] \\
\leq & Z_k 1_{\{\tau \leq k\}} + \rho \frac{1}{\rho^{k+1}} \mathbb{E}[||\boldsymbol{W}_k - \boldsymbol{w}^*||_2^2 | \mathcal{F}_k] 1_{\{\tau > k\}} \\
= & Z_k 1_{\{\tau \leq k\}} + Z_k 1_{\{\tau > k\}} \\
= & Z_k.
\end{aligned}$$

Using the fact that Z_k is a supermartingale we have

$$\begin{aligned}
Z_0 \geq & \mathbb{E}[Z_k | \mathcal{F}_0] \\
\geq & \mathbb{E}[\frac{||\boldsymbol{W}_{k \wedge \tau} - \boldsymbol{w}^*||_2^2}{\rho^{k \wedge \tau}} 1_{k \geq \tau} | \mathcal{F}_0] \\
\geq & \mathbb{E}[\frac{||\boldsymbol{W}_{\tau} - \boldsymbol{w}^*||_2^2}{\rho^{\tau}} 1_{k \geq \tau} | \mathcal{F}_0].
\end{aligned}$$

Now By the definition of stopping time,

$$||\boldsymbol{W}_{\tau} - \boldsymbol{w}^*||_2^2 \geq \epsilon^2 ||\boldsymbol{w}^*||_2^2$$

and using $\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2 \le \epsilon\sqrt{\delta_1}\|\boldsymbol{w}^*\|_2$, we arrive at

$$\epsilon^2\delta_1\|\boldsymbol{w}^*\|_2^2 \ge \mathbb{E}[\frac{\epsilon^2\|\boldsymbol{w}^*\|_2^2}{\rho^\tau}1_{\{k\ge\tau\}}|\mathcal{F}_0].$$

This in turn implies that $\delta_1 \ge \mathbb{E}[\frac{1_{\{k\ge\tau\}}}{\rho^\tau}|\mathcal{F}_0]$. Thus,

$$\delta_1 \ge \mathbb{E}[\frac{1_{\{k\ge\tau\}}}{\rho^\tau}|\mathcal{F}_0] \ge \mathbb{E}[1_{\{k\ge\tau\}}|\mathcal{F}_0] = \mathbb{P}\{k\ge\tau\}.$$

Whence,

$$\delta_1 = \lim_{k\to\infty}\delta_1 \ge \lim_{k\to\infty}\mathbb{P}\{k\ge\tau\} = \mathbb{P}\{\infty > \tau\}.$$

Hence we conclude that $\mathbb{P}\{\tau<\infty\} \le \delta_1 \le 1/2$. Thus

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2 1_{\tau=\infty}] =& \mathbb{E}[\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2|\tau=\infty]\mathbb{P}(\tau=\infty) + 0\cdot\mathbb{P}(\tau<\infty)\\ \ge& \frac{1}{2}\mathbb{E}[\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2|\tau=\infty].\end{aligned}$$

By (6.9) we have

$$\mathbb{E}[\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2|\tau=\infty] \le 2\rho^t\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2.$$

Thus, using Markov's inequality we conclude that

$$\begin{aligned}\mathbb{P}(\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2 > \epsilon\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2|\tau=\infty) \le& \frac{\mathbb{E}[\|\boldsymbol{W}_t - \boldsymbol{w}^*\|_2^2|\tau=\infty]}{\epsilon\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2}\\ \le& \frac{2\rho^t}{\epsilon},\end{aligned}$$

which is bounded by δ_2.

6.1.1 Proof of Lemma 6.2

We use the following identity shown in [23].

$$\mathbb{E}_{I_m(t)}\|\nabla L_{I_m(t)}(\boldsymbol{w}_{t-1})\|_2^2 = \frac{m-1}{m}\|\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\|_2^2 + \frac{1}{m}\mathbb{E}_{I_1(t)}\left[\|\nabla L_{I_1(t)}(\boldsymbol{w}_{t-1})\|_2^2\right] \tag{6.10}$$

To upper bound the first term, using (6.4) with $\boldsymbol{u} = \frac{-\nabla\mathcal{L}(\boldsymbol{w}_{t-1})}{\|\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\|_2}$ we conclude that

$$\begin{aligned}\|\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\|_2 \le& \langle\frac{\nabla\mathcal{L}(\boldsymbol{w}_{t-1})}{\|\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\|_2}, \boldsymbol{w}_{t-1} - \boldsymbol{w}^*\rangle + \frac{1}{4}\|\boldsymbol{w}_{t-1} - w^*\|_2\\ \le& \|\boldsymbol{w}_{t-1} - \boldsymbol{w}^*\|_2 + \frac{1}{4}\|\boldsymbol{w}_{t-1} - w^*\|_2\\ =& \frac{5}{4}\|\boldsymbol{w}_{t-1} - w^*\|_2\end{aligned}$$

holds with probability at least $1 - (n+25)e^{-\gamma n}$.

In order to upper bound the second term on the right hand side of (6.10) we use the following chain of inequalities

$$\begin{aligned}
\mathbb{E}_{I_1(t)}\big[\|\nabla L_{I_1(t)}(\boldsymbol{w}_{t-1})\|_2^2\big] =& \frac{1}{n}\sum_{i=1}^{n}\|\big(\mathrm{ReLU}(\langle \boldsymbol{w}_{t-1}, x_i\rangle) - \mathrm{ReLU}(\langle \boldsymbol{w}^*, x_i\rangle)\big)\|_2^2 \cdot \big(1+\mathrm{sgn}\big(\langle \boldsymbol{w}_{t-1}, x_i\rangle\big)\big)^2\|x_i\|_2^2 \\
\leq& \frac{4}{n}\sum_{i=1}^{n}\|x_i\|_2^2|\langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \boldsymbol{x}_i\rangle|^2 \\
\overset{a}{\leq}& 4\cdot\frac{3d}{2}(\boldsymbol{w}_{t-1}-\boldsymbol{w}^*)^T\frac{1}{n}\sum_{i=1}^{n}x_i x_i^T(\boldsymbol{w}_{t-1}-\boldsymbol{w}^*) \\
\overset{b}{\leq}& 4d\cdot\frac{3d}{2}\cdot\frac{3}{2}\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2 \\
=& 9d\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2
\end{aligned}$$

This holds with probability at least $1-2(n+1)e^{-\gamma d}$. In (a) we used the fact that a high-dimensional Gaussian random vector is well-concentrated on the sphere ofradius $\sqrt{d}$ with high probability and in the last inequality we used a well-known upper bound on the spectral norm of the sample covariance matrix $\|\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^T\| \leq 2$ which holds with high probability.

6.2 Convergence of Quantized SGD (Theorem 3.2)

The proof of this theorem is similar to its counter part in Theorem 3.1. We begin by defining $L_{I_{m_k}(t)}(\boldsymbol{w}) \triangleq \frac{1}{m_k}\sum_{j=1}^{m_k}\ell_{i_t^{(j)}}(\boldsymbol{w})$. We can repeat the argument of the proof of Theorem 3.1 and rewrite (6.2) using the updates (2.2). We begin by taking expectations with respect to the randomness in the quanitzation procedure (denoted by $\mathbb{E}_{Q_s}$). This allows us to conclude that

$$\begin{aligned}
\mathbb{E}_{Q_s}\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2 =& \mathbb{E}_{Q_s}\Bigg[\|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2 - 2\frac{\eta}{K}\langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \sum_{k=1}^{K}Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\rangle \\
&+ \frac{\eta^2}{K^2}\|\sum_{k=1}^{K}Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\|_2^2\Bigg].
\end{aligned}$$

To continue we use a Lemma from [1] which shows the stochastic gradient is unbiased and bounds its variance.

Lemma 6.3 *For any vector $\boldsymbol{v}\in\mathbb{R}^d$,*
(i) $\mathbb{E}[Q_s(\boldsymbol{v})] = \boldsymbol{v}$.
(ii) $\mathbb{E}[\|Q_s(\boldsymbol{v})\|^2] \leq \Big(1+min(\frac{d}{s^2}, \frac{\sqrt{d}}{s})\Big)\|\boldsymbol{v}\|_2^2$.

Using Lemma 6.3, we conclude that

$$\begin{aligned}
\mathbb{E}_{Q_s}\big[\langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \frac{1}{K}\sum_{k=1}^{K}Q_s\big(\nabla L_{I_m(t)}(\boldsymbol{w}_{t-1})\big)\rangle\big] =& \langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \frac{1}{K}\sum_{k=1}^{K}\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\rangle \\
=& \langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \nabla L_{I_m(t)}(\boldsymbol{w}_{t-1})\rangle,
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}_{Q_s}\Big[\|\sum_{k=1}^{K} Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\|_2^2\Big] =& \mathbb{E}_{Q_s}\Bigg(\sum_{k=1}^{K}\|Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\|_2^2 \\
&+ \sum_{i\neq j}\langle Q_s\big(\nabla L_{I_{m_i}(t)}(\boldsymbol{w}_{t-1})\big), Q_s\big(\nabla L_{I_{m_j}(t)}(\boldsymbol{w}_{t-1})\big)\rangle\Bigg) \\
=& \sum_{k=1}^{K}\mathbb{E}_{Q_s}\|Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\|_2^2 \\
&+ \sum_{i\neq j}\langle \nabla L_{I_{m_i}(t)}(\boldsymbol{w}_{t-1}), \nabla L_{I_{m_j}(t)}(\boldsymbol{w}_{t-1})\rangle \\
\leq& \sum_{k=1}^{K}\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)\|\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\|_2^2 \\
&+ \sum_{i\neq j}\langle \nabla L_{I_{m_i}(t)}(\boldsymbol{w}_{t-1}), \nabla L_{I_{m_j}(t)}(\boldsymbol{w}_{t-1})\rangle
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{Q_s,I_m(t)}\Big[\|\sum_{k=1}^{K} Q_s\big(\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\big)\|_2^2\Big] \leq& \sum_{k=1}^{K}\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)\mathbb{E}\|\nabla L_{I_{m_k}(t)}(\boldsymbol{w}_{t-1})\|_2^2 \\
&+ \sum_{i\neq j}\langle \mathbb{E}\left(\nabla L_{I_{m_i}(t)}(\boldsymbol{w}_{t-1})\right), \mathbb{E}\left(\nabla L_{I_{m_j}(t)}(\boldsymbol{w}_{t-1})\right)\rangle \\
\leq& K\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)\left(\frac{9dK}{m}+\frac{25}{16}\right)\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2 \\
&+ (K^2-K)\|\nabla\mathcal{L}(\boldsymbol{w}_{t-1})\|_2^2 \\
\leq& K\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)\left(\frac{9dK}{m}+\frac{25}{16}\right)\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2 \\
&+ \frac{25}{16}(K^2-K)\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2 \\
\leq& K^2\left(\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)(\frac{9d}{m}+\frac{25}{16})+\frac{25}{16}\right)\cdot\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2.
\end{aligned}$$

Whence,

$$\begin{aligned}
\mathbb{E}_{Q_s,I_m(t)}\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2 \leq& \|\boldsymbol{w}_{t-1}-\boldsymbol{w}^*\|_2^2 - 2\eta\langle \boldsymbol{w}_{t-1}-\boldsymbol{w}^*, \nabla\mathcal{L}(\boldsymbol{w}_{t-1})\rangle \\
&+ \eta^2\left(\left(1+\min(\frac{d}{s^2},\frac{\sqrt{d}}{s})\right)(\frac{9d}{m}+\frac{25}{16})+\frac{25}{16}\right)\cdot\|\boldsymbol{w}_t-\boldsymbol{w}^*\|_2^2.
\end{aligned}$$

The remainder of the proof is exactly the same as that of Theorem 3.1.

References

[1] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1709–1720.

[2] F. H. Clark, "Optimization and nonsmooth analysis," *SIAM*.

[3] M. Soltanolkotabi, "Learning relus via gradient descent," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 2007–2017.

[4] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, "Parallel distributed computing using python," *Advances in Water Resources*, vol. 34, no. 9, pp. 1124–1139, 2011.

[5] A. T. Kalai and R. Sastry, "The isotron algorithm: High-dimensional isotonic regression." in *COLT*, 2009.

[6] S. Goel, V. Kanade, A. R. Klivans, and J. Thaler, "Reliably learning the relu in polynomial time," *CoRR*, vol. abs/1611.10258, 2016.

[7] J. Horowitz and W. Härdle, "Direct semiparametric estimation of single-index models with discrete covariates," *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1632–1640, 12 1996.

[8] H. Ichimura, "Semiparametric least squares (sls) and weighted sls estimation of single- index models," Minnesota - Center for Economic Research, Working Papers, 1991.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[10] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 160–167.

[11] S. Oymak, "Stochastic gradient descent learns state equations with nonlinear activations," *arXiv preprint arXiv:1809.03019*, 2018.

[12] G. Wang, G. B. Giannakis, and J. Chen, "Learning relu networks on linearly separable data: Algorithm, optimality, and generalization," *arXiv preprint arXiv:1808.04685*, 2018.

[13] G. Jagatap and C. Hegde, "Learning relu networks via alternating minimization," *arXiv preprint arXiv:1806.07863*, 2018.

[14] X. Zhang, Y. Yu, L. Wang, and Q. Gu, "Learning one-hidden-layer relu networks via gradient descent," *arXiv preprint arXiv:1806.07808*, 2018.

[15] S. Liang, R. Sun, Y. Li, and R. Srikant, "Understanding the loss surface of single-layered neural networks for binary classification," *CoRR*, vol. abs/1803.00909, 2018.

[16] S. Goel and A. Klivans, "Learning depth-three neural networks in polynomial time," *arXiv preprint arXiv:1709.06010*, 2017.

[17] S. Goel, A. Klivans, and R. Meka, "Learning one convolutional layer with overlapping patches," *arXiv preprint arXiv:1802.02547*, 2018.

[18] H. Fu, Y. Chi, and Y. Liang, "Local geometry of one-hidden-layer neural networks for logistic regression," *arXiv preprint arXiv:1802.06463*, 2018.

[19] C. Jin, L. T. Liu, R. Ge, and M. I. Jordan, "Minimizing nonconvex population risk from rough empirical risk," *CoRR*, vol. abs/1803.09357, 2018. [Online]. Available: http://arxiv.org/abs/1803.09357

[20] D. Needell, R. Ward, and N. Srebro, "Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1017–1025.

[21] E. Moulines and F. R. Bach, "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 451–459.

[22] T. Strohmer and R. Vershynin, "A randomized kaczmarz algorithm with exponential convergence," *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, p. 262, Apr 2008.

[23] S. Ma, R. Bassily, and M. Belkin, "The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning," *CoRR*, vol. abs/1712.06559, 2017.

[24] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr, "Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training," in *Advances in Neural Information Processing Systems*, 2018.

[25] C. M. De Sa, C. Zhang, K. Olukotun, C. Ré, and C. Ré, "Taming the wild: A unified analysis of hogwild-style algorithms," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2674–2682.

[26] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1223–1231.

[27] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns," in *Interspeech 2014*, September 2014.

[28] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: compressed optimisation for non-convex problems," *CoRR*, vol. abs/1802.04434, 2018. [Online]. Available: http://arxiv.org/abs/1802.04434

[29] Y. S. Tan and R. Vershynin, "Phase retrieval via randomized kaczmarz: theoretical guarantees," *Information and Inference: A Journal of the IMA*, p. iay005, 2018.

7 Appendix

In the statement of the two Theorems, the probability of success depends on the proximity of the initial estimate to the solution. To alleviate this dependency we directly adapt the *Ensemble* Algorithm used in [29]. See Algorithm 1 for details of this approach.

We now state a result regarding the performance of the ensemble method. We note that the proof of this lemma is essentially identical to its counterpart in [29] and requires only minor modifications.

Proposition 7.1 *(Guarantees for ensemble methods). Consider the setup and assumptions of Theorem 3.1. Furthermore assume that Prob(failure) $\leq 1/3$. Then, for any $\delta' > 0$ there is an absolute constant C such that if $L \geq C \log(1/\delta')$ then the estimate $\hat{\boldsymbol{w}}$ obtained via Algorithm 1 satisfies $||\hat{\boldsymbol{w}} - \boldsymbol{w}^*||_2 \leq 9\epsilon ||\boldsymbol{w}_0 - \boldsymbol{w}^*||_2$.*

Algorithm 1 Ensemble Method

Input: Feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$, labels $y_1, y_2, ..., y_n$, relative error tolerance ϵ, iteration count K to achieve the desired error ϵ, trial count L.

Output: An estimate $\hat{\boldsymbol{w}}$ for $\boldsymbol{w}^*$.

Initialize $\boldsymbol{w}_0$ to satisfy the condition of Theorems 3.1 and 3.2.
for $l = 1, ..., L$ run K SGD updates from the initial point $\boldsymbol{w}_0$ to obtain $\boldsymbol{w}_K^{(l)}$.
for $l = 1, ..., L$, **do**
    if $|B(\boldsymbol{w}_k^{(l)}, 2\sqrt{\epsilon}) \cap \{\boldsymbol{w}_K^{(1)}, ..., \boldsymbol{w}_K^{(L)}\}| \geq L/2$ **then**
        Return $\hat{\boldsymbol{w}} := \boldsymbol{w}_K^{(l)}$