LocalNewton: Reducing Communication Rounds for Distributed Learning

Vipul Gupta[1] **Avishek Ghosh**[1] **Michał Dereziński**[2] **Rajiv Khanna**[2] **Kannan Ramchandran**[1]
Michael W. Mahoney[2]

[1]Department of EECS, University of California, Berkeley
[2]Department of Statistics, University of California, Berkeley

Abstract

To address the communication bottleneck problem in distributed optimization within a master-worker framework, we propose LocalNewton, a distributed second-order algorithm with *local averaging*. In LocalNewton, the worker machines update their model in every iteration by finding a suitable second-order descent direction using only the data and model stored in their own local memory. We let the workers run multiple such iterations locally and communicate the models to the master node only once every few (say L) iterations. LocalNewton is highly practical since it requires only one hyperparameter, the number L of local iterations. We use novel matrix concentration based techniques to obtain theoretical guarantees for LocalNewton, and we validate them with detailed empirical evaluation. To enhance practicability, we devise an adaptive scheme to choose L, and we show that this reduces the number of local iterations in worker machines between two model synchronizations as the training proceeds, successively refining the model quality at the master. Via extensive experiments using several real-world datasets with AWS Lambda workers and an AWS EC2 master, we show that LocalNewton requires fewer than 60% of the communication rounds (between master and workers) and less than 40% of the end-to-end running time, compared to state-of-the-art algorithms, to reach the same training loss.

1 INTRODUCTION

An explosion in data generation and data collection capabilities in recent years has resulted in the segregation of computing and storage resources. Distributed machine learning is one example where each worker machine processes only a subset of the data, while the master machine coordinates with workers to learn a good model. Such coordination can be time-consuming since it requires frequent communication between the master and worker nodes, especially for systems that have large compute resources, but are bottlenecked by communication costs.

Communication costs in distributed optimization can be broadly classified into two types—(a) latency cost and (b) bandwidth cost [Demmel, 2013]. Latency is the fixed cost associated with sending a message, and it is generally independent of the size of the message. Bandwidth cost, on the other hand, is directly proportional to the size of the message. Many recent works have focused on reducing the bandwidth cost by reducing the size of the gradient or the model to be communicated using techniques such as sparsification [Acharya et al., 2019, Stich et al., 2018], sketching [Ivkin et al., 2019, Konečnỳ et al., 2016] and quantization [Ghosh, Avishek et al., 2020, Lin et al., 2017, Bernstein et al., 2018, Dong et al., 2019, Shen et al., 2020]. Schemes that perform inexact updates in each iteration, however, can *increase* the number of iterations required to converge to the same quality model, both theoretically and empirically [Acharya et al., 2019, Mayekar and Tyagi, 2020]. This can, in turn, increase the total training time in systems where latency costs dominate bandwidth costs.

One setting where latency costs may outweigh bandwidth costs is federated learning, where the computation is performed locally at the mobile device (which is generally the source of the data) due to a high-cost barrier in transferring the data to traditional computing platforms [Konečnỳ et al., 2016]. Such mobile resources (e.g., mobile phones, wearable devices, etc.) have reasonable compute power, but they can be severely limited by communication latency (e.g., inadequate network connection to synchronize frequent model updates). For this reason, schemes like Local Stochastic Gradient Descent (Local SGD) have become popular, since they try to mitigate the communication costs by performing more *local* computation at the worker machines, thus substantially reducing the number of communication rounds

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:632–642.

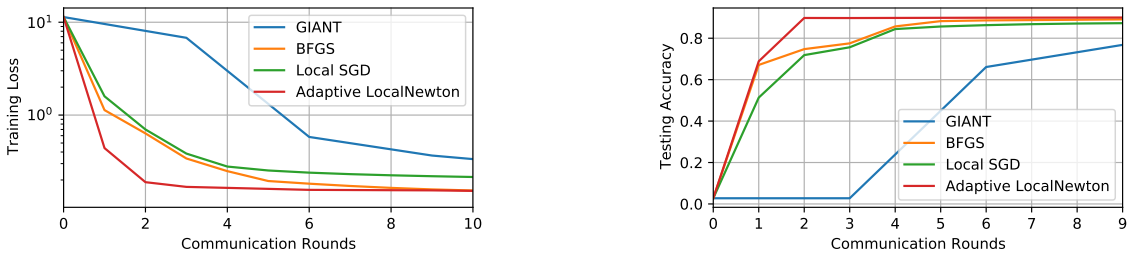


Figure 1: Training loss and testing accuracy versus communication rounds for Adaptive LocalNewton and existing schemes for communication-efficient optimization.

required [McMahan et al., 2017].

Serverless systems—such as Amazon Web Services (AWS) Lambda and Microsoft Azure Functions—are yet another example where high communication latency between worker machines dominates the running time of the algorithm [Hellerstein et al., 2018]. Such systems are gaining popularity due to the ease-of-management, greater elasticity and high scalability [Jonas et al., 2019]. These systems use cloud storage (e.g., AWS S3) to store enormous amounts of data, while using a large number of low-quality workers for large-scale computation. Naturally, the communication between the high-latency storage and the commodity workers is extremely slow [e.g., see Jonas et al., 2017], resulting in impractical end-to-end times for many popular optimization algorithms such as SGD [Hellerstein et al., 2018]. Furthermore, communication failures between the cloud storage and serverless workers consistently give rise to stragglers, and this introduces synchronization delays [Gupta et al., 2020a].

These trends suggest that optimization schemes that reduce communication rounds between workers are highly desirable. In this paper, we go one-step forward—we propose and analyze a second order method with *local* computations. Being a second order algorithm, the iteration complexity of LocalNewton is inherently low. Moreover, its local nature further cuts down the communication cost between the worker and the master node. To the best of our knowledge, this is the first work to propose and analyze a second order optimization algorithm with local averaging.

Additionally, in this paper, we introduce an adaptive variant of the LocalNewton algorithm, namely Adaptive LocalNewton. This algorithm chooses the number of local iterations adaptively at the master after each communication round by observing the change in training loss. Thus, it further refines the iterates obtained through LocalNewton by adaptively and successively reducing the number of local iterations, thereby improving the quality of the model updates. Furthermore, when the number of local iterations, L, reduces to 1, Adaptive LocalNewton automatically switches to a standard second order optimization algorithm, namely GIANT, proposed in [Wang et al., 2018].

In Sec. 4, we show that the iterates of LocalNewton converge to a norm ball (with small radius) around the global minima. From this point of view, we may think of LocalNewton as an initialization algorithm, rather than an optimization one; since it takes the iterates close to the optimal point in only a few rounds of communication. After reaching sufficiently close to the solution point, one may choose some standard optimization algorithm to reach to the solution. In Adaptive LocalNewton, we first exploit the fast convergence of LocalNewton ($L \geqslant 1$), and afterwards, when $L = 1$, Adaptive LocalNewton switches to the standard optimization algorithm (e.g., GIANT [Wang et al., 2018]).

Our contributions. Inspired by recent progress in local optimization methods (that reduce communication cost by limiting the frequency of synchronization) and distributed and stochastic second order methods (that use the local curvature information), we propose a local second-order algorithm called LocalNewton. The proposed LocalNewton method saves on communication costs in two ways. First, it updates the models at the master only sporadically, thus requiring only one communication round per multiple iterations. Second, it uses the second-order information to reduce the number of iterations, and hence it reduces the overall rounds of communication.

Important features of LocalNewton include:

1. *Simplicity*: In LocalNewton, each worker takes only a few Newton steps [Boyd and Vandenberghe, 2004] on local data, agnostic of other workers. These local models are then averaged once every $L(\geqslant 1)$ iterations at the master node.

2. *Practicality*: Unlike many first-order and distributed second-order schemes, LocalNewton does not require hyperparameter tuning for step-size, mini-batch size, etc., and the only hyperparameter required is the number of local iterations L. We also propose Adaptive LocalNewton, an adaptive version which automatically reduces L as the training proceeds by monitoring the training loss at the master.

3. *Convergence guarantees*: In general, proving convergence guarantees for local algorithms is not straightforward. Only recently, it has been proved [Stich, 2018] that local SGD converges as fast as SGD, thereby explaining the well-studied empirical successes [Konečnỳ et al., 2016]. In this paper,

we develop novel techniques to highlight the convergence behaviour of LocalNewton.

4. *Reduced training times*: We implement LocalNewton on the Pywren framework [Jonas et al., 2017] using AWS Lambda[1] workers and an AWS EC2[2] master. Through extensive empirical evaluation, we show that the significant savings in terms of communication rounds translate to savings in running time on this high-latency distributed computing environment.

5. *Adaptivity*: We propose Adaptive LocalNewton, which is an adaptive variant of the LocalNewton algorithm. In the adaptive scheme, based on the change in function objective value, the master modulates the number of local iterations at the worker machines. This improves the quality of the model updates as discussed in detail in Sec. 3.2.

6. *LocalNewton as an initialization algorithm*: Since the convergence guarantees of LocalNewton (see Sec. 4) only ensure that the iterates stay in a norm ball around the minima, one may rethink LocalNewton as an initialization algorithm, rather than an optimization one. In only a very few communication rounds, LocalNewton takes the iterates very close to the optimal solution. After that, our algorithm switches to a standard second-order algorithm, GIANT [Wang et al., 2018].

Fig. 1 illustrates savings due to Adaptive LocalNewton, where we plot training loss and test accuracy with communication rounds, for several popular communication-efficient schemes for logistic regression on the w8a dataset [Chang and Lin, 2011] (see Sec. 5 for a details on experiments). Observe that Adaptive LocalNewton reaches close to the optimal training loss very quickly, when compared to schemes like Local SGD [Konečnỳ et al., 2016, Stich, 2018], GIANT [Wang et al., 2018] and BFGS [Fletcher, 2013].

Related Work. In recent years, schemes such as local SGD have gained popularity, as they are communication-efficient due to only sporadic model updates at the master [Konečnỳ et al., 2016, McMahan et al., 2017]. Such schemes that show great promise have eluded a thorough theoretical analysis until recently, when it was shown that local SGD converges at the same rate as mini-batch SGD [Stich, 2018, Haddadpour et al., 2019, Dieuleveut and Patel, 2019]. Similar ideas that reduce communication by averaging the local models sporadically have also been applied in training neural networks to improve the training times and/or model performance [Lin et al., 2020, Gupta et al., 2020b]. Apart from local first-order methods, many distributed second-order (also known as Newton-type) algorithms have been proposed to reduce communication (e.g., Jadbabaie et al. [2009] and Tutunov et al. [2019] propose second-order algorithms in a decentralized setting where communication happens over a predefined graph). More recently, many communication-efficient second-order methods were proposed in the centralized setting, e.g., see [Wang et al., 2018, Shamir et al., 2014, Zhang and Lin, 2015, Smith et al., 2016, Reddi et al., 2015, Duenner et al., 2018, Dereziński and Mahoney, 2019, Dereziński et al., 2020, Ghosh et al., 2020a,b]. Such methods use both the gradient and the curvature information to provide an order of improvement in convergence, compared to vanilla first-order methods. This is done at the cost of more local computation per iteration, which is acceptable for systems with high communication latency. However, such algorithms require at least two communication rounds (for averaging gradients and the second-order descent direction), and a thorough knowledge of a fundamental trade-off between communication and local computation is still lacking for these methods. Stochastic second order optimization theory has been developed recently [Roosta-Khorasani and Mahoney, 2016a,b, Xu et al., 2020], and second order implementations motivated by this theory have been shown to outperform state-of-the-art [Yao et al., 2019, 2020].

[1]A high-latency *serverless* computing platform.

[2]A traditional *serverful* computing platform.

2 PROBLEM FORMULATION

We first define the notation used and then introduce the basic problem setup considered in this paper.

Notation. Throughout the paper, vectors (e.g., $\mathbf{g}$) and matrices (e.g., $\mathbf{H}$) are represented as bold lowercase and uppercase letters, respectively. For a vector $\mathbf{g}$, $\|\mathbf{g}\|$ denotes its ℓ_2 norm, and $\|\mathbf{H}\|_2$ denotes the spectral norm of matrix $\mathbf{H}$. The identity matrix is denoted as $\mathbf{I}$, and the set $\{1, 2, \cdots, n\}$ is denoted as $[n]$, for all positive integers n. Further, we use superscript (e.g., $\mathbf{g}^k$) to denote the worker index and subscript (e.g., $\mathbf{g}_t$) to denote the iteration counter (i.e., time index), unless stated otherwise.

Problem Setup. We are interested in solving empirical risk minimization problems of the following form in a distributed fashion

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left\{ f(\mathbf{w}) \triangleq \frac{1}{n}\sum_{j=1}^{n} f_j(\mathbf{w}) \right\}, \tag{1}$$

where $f_j(\cdot) : \mathbb{R}^d \to \mathbb{R}$, for all $j \in [n] = \{1, 2, \cdots, n\}$, models the loss of the j-th observation given an underlying parameter estimate $\mathbf{w} \in \mathbb{R}^d$. In machine learning, such problems arise frequently, e.g., logistic and linear regression, support vector machines, neural networks and graphical models. Specifically, in the case of logistic regression,

$$f_j(\mathbf{w}) = \ell_j(\mathbf{w}^T\mathbf{x}_j) = \log(1 + e^{-y_j\mathbf{w}^T\mathbf{x}_j}) + \frac{\gamma}{2}\|\mathbf{w}\|^2,$$

where $\ell_j(\cdot)$ is the loss function for sample $j \in [n]$ and γ is an appropriately chosen regularization parameter. Also, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ is the sample matrix containing the input feature vectors $\mathbf{x}_j \in \mathbb{R}^d, j \in [n]$, and $\mathbf{y} = [y_1, y_2, \cdots, y_n]$ is the corresponding label vector.

Hence, $(\mathbf{x}_j, y_j)$ together define the j-th observation and $(\mathbf{X}, \mathbf{y})$ define the training dataset.

For such problems, the gradient and the Hessian at the t-th iteration are given by

$$\mathbf{g}_t = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\mathbf{w}_t) \text{ and } \mathbf{H}_t = \frac{1}{n}\sum_{j=1}^{n} \nabla^2 f_j(\mathbf{w}_t),$$

respectively, where $\mathbf{w}_t$ is the model estimate at the t-th iteration.

Data distribution at each worker: Let there be a total of K workers. We assume that the k-th worker is assigned a subset $\mathcal{S}_k \subset [n]$, for all $k \in [K] = \{1, 2, \cdots, K\}$, of the n data points, chosen uniformly at random without replacement.[3] Let the number of samples at each worker be $s = |\mathcal{S}_k| \ \forall \ k \in [K]$, where $s \ll n$ in practice. Also, by the virtue of sampling without replacement, we have $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_K = [n]$ and $\mathcal{S}_i \cap \mathcal{S}_j = \Phi$ for all $i, j \in [K]$. Hence, the number of workers is given by $K = n/s$.

3 ALGORITHMS

In this section, we propose two novel algorithms for distributed optimization. First, we propose LocalNewton, a second order algorithm with local averaging. Subsequently, we also propose an adaptive variant of LocalNewton. Here LocalNewton acts as a good initialization scheme that pushes its iterates close to the optimal solution in a small number of communication rounds. Finally, a standard second-order algorithm is used to converge to the optimal solution.

3.1 LOCALNEWTON

We consider synchronous second-order methods for distributed learning, where local models are synced after every L iterations. Let $\mathcal{I}_t \subseteq [t]$ be the set of indices where the model is synced, that is, $\mathcal{I}_t = [0, L, 2L, \cdots, t_0]$, where t_0 is the last iteration just before t where the models were synced.

At the k-th worker in the t-th iteration, the local function value (at the local iterate $\mathbf{w}_t^k$) is

$$f^k(\mathbf{w}_t^k) = \frac{1}{s}\sum_{j \in \mathcal{S}_k} f_j(\mathbf{w}_t^k). \tag{2}$$

The k-th worker tries to minimize the local function value in Eq (2) in each iteration. The corresponding local gradient $\mathbf{g}_t^k$ and local Hessian $\mathbf{H}_t^k$, respectively, at k-th worker in t-th iteration can be written as

$$\mathbf{g}_t^k = \frac{1}{s}\sum_{j \in \mathcal{S}_k} \nabla f_j(\mathbf{w}_t^k) \text{ and } \mathbf{H}_t^k = \frac{1}{s}\sum_{j \in \mathcal{S}_k} \nabla^2 f_j(\mathbf{w}_t^k).$$

Let us consider the following LocalNewton update at the k-th worker and $(t+1)$-st iteration:

$$\mathbf{w}_{t+1}^k = \begin{cases} \mathbf{w}_t^k - \alpha_t^k \mathbf{H}^k(\mathbf{w}_t^k)^{-1}\mathbf{g}^k(\mathbf{w}_t^k), & \text{if } t \notin \mathcal{I}_t \\ \bar{\mathbf{w}}_t - \alpha_t^k \mathbf{H}^k(\bar{\mathbf{w}}_t)^{-1}\mathbf{g}^k(\bar{\mathbf{w}}_t), & \text{if } t \in \mathcal{I}_t, \end{cases} \tag{3}$$

where $\bar{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_t^k \ \forall \ t$, and α_t^k is the step-size at the k-th worker at iteration t.[4]

Also, define the local descent direction at the k-th worker at iteration t as $\mathbf{p}_t^k = \alpha_t^k \mathbf{H}^k(\mathbf{w}_t^k)^{-1}\mathbf{g}^k(\mathbf{w}_t^k)$ and similarly define $\bar{\mathbf{p}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{p}_t^k$. We can see that $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \bar{\mathbf{p}}_t$. Detailed steps for LocalNewton are provided in Algorithm 1.

Note that $\bar{\mathbf{w}}_t$ is not explicitly calculated for all t, but only for $t \in \mathcal{I}_t$. However, we will use the technique of perturbed iterate analysis and show the convergence of the sequence $f(\bar{\mathbf{w}}_1), f(\bar{\mathbf{w}}_2), \cdots, f(\bar{\mathbf{w}}_t)$ to $f(\mathbf{w}^*)$. In the next section, we present Adaptive LocalNewton, an algorithm to adaptively choose the number of local iterations, L.

Algorithm 1: LocalNewton

Input : Local function $f^k(\cdot)$ at the k-th worker; Initial iterate $\bar{\mathbf{w}}_0 \in \mathbb{R}^d$; Line search parameter $0 < \beta \leqslant 1/2$; Number of iterations T, Set $\mathcal{I}_T \subseteq \{1, 2, \cdots, T\}$ where models are synced

for $k = 1$ *to* K *in parallel* **do**
  Initialization: $\mathbf{w}_0^k = \bar{\mathbf{w}}_0$
  for $t = 0$ *to* $T - 1$ **do**
    if $t \in \mathcal{I}_T$ **then**
      `// Master averages the local models`
      $\bar{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_t^k$
      $\mathbf{w}_t^k = \bar{\mathbf{w}}_t$
    end
    `// Compute the local gradient`
    $\mathbf{g}^k(\mathbf{w}_t^k) = \nabla f^k(\mathbf{w}_t^k)$.
    `// Compute the update direction`
    $\mathbf{p}_t^k = \mathbf{H}^k(\mathbf{w}_t^k)^{-1}\mathbf{g}^k(\mathbf{w}_t^k)$
    Find step-size α_t^k using line search (Eq. (5))
    Update model: $\mathbf{w}_{t+1}^k = \mathbf{w}_t^k - \alpha_t^k \mathbf{p}_t^k$
  end
end

[3] This corresponds simply to partitioning the dataset and assigning an equal number of observations to each worker, if the observations are independent and identically distributed. If not, randomly shuffling the observations and then performing a data-independent partitioning is equivalent to uniform sampling without replacement.

[4] In practice, one need not calculate the exact $\mathbf{H}^k(\mathbf{w}_t^k)^{-1}\mathbf{g}^k(\mathbf{w}_t^k)$, and efficient algorithms like conjugate gradient descent can be used [Shewchuk, 1994].

Algorithm 2: Adaptive LocalNewton

Input : Minimum decrement $\delta(>0)$ in global loss function;

Initialization: $f_{prev} = f(\bar{\mathbf{w}}_0)$, Number of local iterations L, Set $\mathcal{I}_T \subseteq \{1, 2, \cdots, T\}$ where models are synced every L local iterations
for $t = 1$ *to* $T-1$ **do**
  // k-th worker runs LocalNewton locally to get $\mathbf{w}_t^k$
  Workers run Algorithm 1
  if $t \in \mathcal{I}_T$ **then**
    // Master averages the local models
    $\bar{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_t^k$
    if $f_{prev} - f(\bar{\mathbf{w}}_t) < \delta$ *and* $L \geqslant 1$ **then**
      // The global function did not decrease enough
      if $L = 1$ **then**
        Switch to GIANT [Wang et al., 2018]
      end
      else
        Decrease L: $L = L - 1$
        Update $\mathcal{I}_T$ according to the new value of L
      end
    end
    $f_{prev} = f(\bar{\mathbf{w}}_t)$
  end
end

3.2 ADAPTIVE LOCALNEWTON

Motivating Example (Least-squares). Let us now consider the simple example of unregularized linear least squares, i.e., the loss function at the k-th worker is $f^k(\mathbf{w}) = \frac{1}{s}\|\mathbf{y}^k - \mathbf{X}^k\mathbf{w}\|^2$, where $\mathbf{y}^k = [y_1^k, y_2^k, \ldots, y_s^k]^\top \in \mathbb{R}^s$ and $\mathbf{X} = [\mathbf{x}_1^k, \ldots, \mathbf{x}_s^k]^\top \in \mathbb{R}^{s\times d}$. Note that one second-order iteration (with step-size one) reaches the optimal solution, say $(\mathbf{w}^k)^* = \mathbf{w}_t^k - (\mathbf{H}_t^k)^{-1}\mathbf{g}^k\mathbf{w}_t^k = [(\mathbf{X}^k)^T\mathbf{X}^k]^{-1}(\mathbf{X}^k)^T\mathbf{y}^k \;\forall\; \mathbf{w}_t^k \in \mathbb{R}^d$, for the local loss function $f^k(\mathbf{w})$ at the k-th worker.

Thus, applying LocalNewton here with $L \geqslant 1$ would imply that the local iterates at the k-th worker are fixed at $\mathbf{w}_t^k = (\mathbf{w}^k)^*$ while the global iterates at the master are fixed at $\bar{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^{K}(\mathbf{w}^k)^*$ for all $t \in [T]$. Note that $\bar{\mathbf{w}}_t \neq \bar{\mathbf{w}}^*$ in general, where $\bar{\mathbf{w}}^*$ is the optimal solution for the global problem in Eq. (1). Hence, LocalNewton (Algorithm 1) does not reach the optimal solution for unregularized least-squares. In fact, in Theorems 4.3 and 4.4, we show that running LocalNewton algorithm results in an error floor of the order $1/\sqrt{s}$ for any convex loss function.

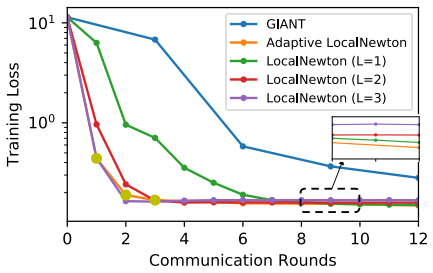
Motivated from the above example, if we want to attain the optimal solution, one needs to switch to optimization algorithms which yield no error floor. One standard example of such an algorithm is GIANT (Wang et al. [2018]), which is a communication-efficient distributed second-order algorithm. However, for general convex functions, the convergence of GIANT requires the initial point to be close to the optimal solution. From this point of view, one can think of LocalNewton as an initialization scheme that pushes the iterates close to the solution (within a radius of $\mathcal{O}(1/\sqrt{s})$) with a few rounds of communication. Then, one can switch to the GIANT algorithm to obtain convergence to the solution point. We call this algorithm Adaptive LocalNewton, where the master modulates the value of L successively over iterations and finally switches to GIANT to obtain the final solution. The details are given in Algorithm 2.

Recall that GIANT synchronizes the local gradients and the local descent direction in every iteration [Wang et al., 2018]. Further, it finds the step-size by doing a distributed backtracking line-search requiring an additional round of communication (Sec. 5.2, Wang et al. [2018]). Finally, the master updates the model by using the average descent direction and the obtained step-size and ships the model to all the workers. Thus, each iteration in GIANT requires three rounds of communication. This approach has compared favorably to other popular distributed second-order methods (e.g., DANE Shamir et al. [2014], AGD Nesterov [2014], BFGS Fletcher [2013], CoCoA Smith et al. [2016], DiSCO Zhang and Lin [2015]).

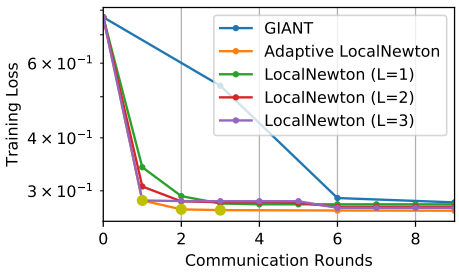
In Fig. 2, we compare GIANT to LocalNewton, where LocalNewton is run with 100 workers for $L = 1, 2$ and 3, for two datasets—w8a and EPSILON obtained from LIBSVM [Chang and Lin, 2011]. Note that LocalNewton converges much faster with respect to communication rounds for all the three datasets since it communicates intermittently, i.e., once every few local second-order iterations (e.g., after 3 local iterations for $L = 3$). Not shown here is that testing accuracy follows the same trends. Further, the quality of the final solution improves as we reduce L. However, it reaches extremely close to the optimal training loss, but it converges very slowly (or flattens out) after that.

These empirical observations further motivate Adaptive LocalNewton: a second-order distributed algorithm that adapts the number of local iterations as the training progresses and ultimately finishes with GIANT. This can be done by monitoring the objective function at the master, e.g., reduce L if the loss stops improving (or switch to GIANT if $L = 1$).[5] See Algorithm 2, where we provide the pseudo-code for Adaptive LocalNewton. Whenever the global function value at the master does not decrease more than a constant δ, we

[5] To further reduce the communication rounds and dependency on L, each worker can update the model for multiple values of L and send the concatenated model updated to the master. The master can decide the right value of L by evaluating the loss/accuracy for these different models.



(a) w8a dataset



(b) EPSILON dataset

Figure 2: Comparing LocalNewton (for different values of L) and GIANT. In general, LocalNewton reaches very close to the optimal solution. In that region, however, the convergence rate of LocalNewton is slow. This is mitigated by using Adaptive LocalNewton which appends the LocalNewton iterations with better quality (but more expensive) updates from GIANT.

decrease the value of L to improve the quality of second order estimate. In this sense, Adaptive LocalNewton can be seen as providing a carefully-constructed or gradually-annealed initialization for GIANT.

Comparison with GIANT: Algorithm 2 is further motivated by the theoretical guarantees we obtain in the subsequent section. In Theorems 4.3 and 4.4 of Sec. 4, we prove the convergence of LocalNewton to the optimal solution within an error floor starting with any initial point in $\mathbb{R}^d$. In sharp contrast, Theorem 2 in GIANT [Wang et al., 2018], the authors convergence guarantees to the optimal solution when the current model is sufficiently close to the optimal model. From Fig. 2, we see that Adaptive LocalNewton significantly outperforms GIANT in terms of rounds of communication. Adaptive LocalNewton starts from L=3, and yellow dots in its plot denote the reduction in the value of L by one or a switch to GIANT if $L = 1$.

4 CONVERGENCE GUARANTEES

In this section, we present the main theoretical contributions of the paper. For this, we only consider the (non-adaptive) LocalNewton algorithm. Obtaining theoretical guarantees for Adaptive LocalNewton is kept as an interesting future work.

First, we delineate some assumptions on $f(\cdot)$ required to prove theoretical convergence of the proposed method.

Assumptions: We make the following standard assumptions on the objective function $f(\cdot)$ for all $\mathbf{w} \in \mathbb{R}^d$:

1. $f_i(\cdot)$, for all $i \in [n]$, is twice differentiable.
2. $f(\cdot)$ is κ-strongly convex, that is, $\nabla^2 f(\mathbf{w}) \succcurlyeq \kappa \mathbf{I}$.
3. $f(\cdot)$ is M-smooth, that is, $\nabla^2 f(\mathbf{w}) \preccurlyeq M\mathbf{I}$.
4. $\|\nabla^2 f_i(\cdot)\|_2, i \in [n]$, is upper bounded. That is, $\nabla^2 f_i(\mathbf{w}) \preccurlyeq B\mathbf{I}$, for all $i \in [n]$.

In the following lemma, we make use of matrix concentration inequalities to show that, for sufficiently large sample size s, the local Hessian at each worker is also strongly convex and smooth with high probability. The proof of all auxiliary lemmas is deferred to Appendix 1.

Lemma 4.1. *Let $f(\cdot)$ satisfy assumptions 1-4 and $0 < \epsilon \leqslant 1/2$ and $0 < \delta < 1$ be fixed constants. Then, if $s \geqslant \frac{4B}{\kappa\epsilon^2} \log \frac{2d}{\delta}$, the local Hessian at the k-th worker satisfies*

$$(1-\epsilon)\kappa \preccurlyeq \nabla^2 f^k(\mathbf{w}) = \mathbf{H}^k(\mathbf{w}) \preccurlyeq (1+\epsilon)M, \quad (4)$$

for all $\mathbf{w} \in \mathbb{R}^d$ and $k \in [K]$ with probability at least $1-\delta$.

Step-size selection: Let each worker locally choose a step-size according to the following rule

$$\alpha_t^k = \max_{\alpha \leqslant \alpha^\star} \alpha \quad \text{such that}$$
$$f^k\left(\mathbf{w}_t^k - \alpha \mathbf{p}_t^k\right) \leqslant f^k\left(\mathbf{w}_t^k\right) - \alpha\beta(\mathbf{p}_t^k)^T \nabla f^k\left(\mathbf{w}_t^k\right), \quad (5)$$

for some constant $\beta \in (0, 1/2]$, where the parameter $\alpha^\star (\leqslant 1)$ depends on the properties of the objective function:[6]

$$\alpha^\star \leqslant \min\left\{\frac{(1-\beta)\kappa}{M}, \frac{2\beta\kappa^2}{3M[M-\kappa/4]}\right\}. \quad (6)$$

Now, we are almost ready to prove the main theorems—Theorem 4.3 discusses the case when $L = 1$, and Theorem 4.4 discusses the case when $L > 1$. Before that, let us state the following auxiliary lemma which is required to prove the main theorems.

Lemma 4.2. *Let the function $f(\cdot)$ satisfy assumptions 1-3, and suppose that step-size α_t^k satisfies the line-search condition in (5). Also, let $0 < \epsilon < 1/2$ and $0 < \delta < 1$ be fixed constants. Moreover, let the sample size $s \geqslant$*

[6] We introduce $\alpha^\star$ here to establish theoretical guarantees. In our empirical results, we use the Armijo backtracking line-search rule with $\alpha^\star = 1$ (e.g., see Boyd and Vandenberghe [2004]) to find the right step-size.

$\frac{4B}{\kappa\epsilon^2}\log\frac{2d}{\delta}$. *Then, the LocalNewton update, defined in Eq.* (3)*, at the* k*-th worker satisfies*

$$f^k(\mathbf{w}_{t+1}^k) - f^k(\mathbf{w}_t^k) \leqslant -\psi||\mathbf{g}_t^k||^2 \;\forall\; k \in [K],$$

with probability at least $1-\delta$*, where* $\psi = \frac{\alpha^\star\beta}{M(1+\epsilon)}$.

We next use the result in Lemma 4.2 to prove linear convergence for the global function $f(\cdot)$. We first prove guarantees for the $L = 1$ case, where the models are communicated every iteration but the gradient is computed locally instead of globally contrary to previous results [Wang et al., 2018] (thus reducing two communication rounds per iteration). We then extend it to the general case of $L > 1$ and show that the updates converge at a sublinear rate in that case.

Theorem 4.3. *[*$L = 1$ *case] Suppose Assumptions 1-5 hold and the step-size* α_t^k *satisfies the line-search condition* (5)*. Also, let* $0 < \delta < 1$, $0 < \epsilon, \epsilon_1 < 1/2$ *be fixed constants and let* $\Gamma = \max_{1\leqslant i\leqslant n}\|\nabla f_i(.)\|$*. Moreover, assume that the sample size for each worker satisfies* $s \geqslant \frac{4B}{\kappa\epsilon^2}\log\frac{2dK}{\delta}$*, where the samples are chosen without replacement. Then, with the LocalNewton updates,* $\{\bar{\mathbf{w}}_t\}_{t\geqslant 0}$*, from Algorithm 1 and* $L = 1$*, we obtain*

1. *If* $s \gtrsim \frac{\Gamma^2}{\epsilon_1^2 G^2}\log(d/\delta)$ *for* $G = \min_k \|g^k(\bar{\mathbf{w}}_t)\|$*, we get with probability at least* $1 - 6K\delta$*,*

$$f(\bar{\mathbf{w}}_{t+1}) - f(\mathbf{w}^*) \leqslant \rho_1(f(\bar{\mathbf{w}}_t) - f(\mathbf{w}^*)).$$

2. *We obtain, with probability at least* $1 - 6K\delta$*,*

$$f(\bar{\mathbf{w}}_{t+1}) - f(\mathbf{w}^*) \leqslant \rho_2(f(\bar{\mathbf{w}}_t) - f(\mathbf{w}^*)) + \eta\cdot\frac{\Gamma}{\kappa(1-\epsilon)},$$

where $\eta = \frac{1}{\sqrt{s}}\Gamma(1+\sqrt{2\log(\frac{1}{\delta})})$.

Here $\rho_i = (1-2\kappa C_i)$*, for* $i = \{1,2\}$, $C_1 = \frac{(1-\epsilon)\psi}{2} - \frac{\epsilon_1}{\kappa(1-\epsilon)}$, $C_2 = \frac{\psi(1-\epsilon)}{2}$*, and* $\psi = \frac{\alpha^\star\beta}{M(1+\epsilon)}$.

Proof. The proof is presented in Appendix 2. Here, we provide a sketch of the proof.

1. Due to the uniform sampling guarantee from Lemma 4.1, the strong-convexity and smoothness of the global function $f(\cdot)$ implies that the local function at the k-th worker, $f^k(\cdot)$, also satisfy similar properties. Using this, we can lower bound $f(\bar{\mathbf{w}}_t) - f(\bar{\mathbf{w}}_{t+1})$ in terms of $\frac{1}{K}\sum_{k=1}^K f^k(\mathbf{w}_t^k) - f^k(\mathbf{w}_{t+1}^k)$.
2. Apply Lemma 4.2 (that is, the result for standard Newton step) which says $f^k(\mathbf{w}_t^k) - f^k(\mathbf{w}_{t+1}^k) \geqslant \psi||\mathbf{g}_t^k||^2 \;\forall\; k \in [K]$.
3. Using uniform sketching argument, local gradients $g^k(\bar{\mathbf{w}}_t)$ are close to global gradient $g(\bar{\mathbf{w}}_t)$.

□

Some remarks regarding the convergence guarantee in Theorem 4.3 are in order.

Remark 1. *The above theorem implies that for* $L = 1$*, the convergence rate of LocalNewton is linear with high probability. Choosing* $\delta = 1/\mathsf{poly}(K)$*, we obtain the high probability as* $1 - 1/\mathsf{poly}(K)$.

Remark 2. *We have two different settings in the above theorem. The first setting implies that provided the local gradients* $\{g^k(\bar{\mathbf{w}}_t)\}_{k=1}^K$ *are large enough, and the amount of local data* s *is reasonably large, then the convergence is* purely *linear and does not suffer an error floor. This will typically happen in the earlier iterations of LocalNewton. Note that the gradients vanish as we get closer to the optimum, which is why the setting will eventually be violated. If this happens, we move to the next setting.*

Remark 3. *The second setting implies that, if the gradient condition and the restriction of* s *are violated, although the convergence rate of LocalNewton is still linear, the algorithm incurs an error floor. However, in this setting, the error floor is* $\mathcal{O}(1/\sqrt{s})$*, and hence it is quite small for sufficiently large sample-size,* s*, at each worker.*

To understand the linear convergence rate of LocalNewton, we consider the following example. Assume all the workers initialize at $\bar{\mathbf{w}}_0$ and run LocalNewton with $L = 1$ for T iterations and the first setting is true (that is, $\|\mathbf{g}^k(\mathbf{w}_t)\| \geqslant G$ for all $k \in [K]$ and $t \in [T]$). Then, from Theorem 4.3, to reach within ξ of the optimal function value (that is, $f(\bar{\mathbf{w}}_T) - f(\bar{\mathbf{w}}^*) \leqslant \xi$), the number of iterations T is upper bounded by

$$T \leqslant \left(\log\frac{1}{\rho_1}\right)\log\frac{\xi}{f(\bar{\mathbf{w}}_0) - f(\bar{\mathbf{w}}^*)}$$

with probability $1 - 6K\delta$ for a sample size

$$s \geqslant \max\left\{\frac{4B}{\kappa\epsilon^2}\log\frac{2dKT}{\delta}, \frac{\Gamma^2}{\epsilon_1^2 G^2}\log\frac{dT}{\delta}\right\}.$$

(Note the increase in sample size s by a factor of T in the $\log(\cdot)$ due to a union bound). The fully synchronized second order method GIANT [Wang et al., 2018] also has similar linear quadratic convergence but it assumes that the gradients are synchronized in every iteration. We remove this assumption by tracking how far the iterate deviates when the gradients are computed locally, thereby cutting the communication costs in half while still showing linear convergence (within some error floor in the most general case).

We now prove convergence guarantees for the case when $L > 1$ in this algorithm.

Theorem 4.4 ($L \geqslant 1$ case)**.** *Suppose Assumptions 1-4 hold and step-size* α_t^k *solves the line-search condition* (5)*. Also, let* $0 < \delta < 1$, $0 < \epsilon < 1/2$ *be fixed constants and let*

$\Gamma = \max_{1 \leqslant i \leqslant n} \|\nabla f_i(.)\|$. *Moreover, assume that the sample size for each worker satisfies* $s \geqslant \frac{4B}{\kappa\epsilon^2} \log \frac{2dK}{\delta}$, *where the samples are chosen without replacement. Then, the LocalNewton updates,* $\{\bar{\mathbf{w}}_t\}_{t \geqslant 0}$, *from Algorithm 1 and* $L \geqslant 1$, *with probability at least* $1 - 6LK\delta$, *satisfy*

$$f(\bar{\mathbf{w}}_{t+1}) - f(\bar{\mathbf{w}}_{t_0}) \leqslant -C \sum_{\tau=t_0}^{t} \left(\frac{1}{K} \sum_{k=1}^{K} \|\mathbf{g}_\tau^k\|^2 \right) + \eta \cdot \frac{L\Gamma}{\kappa(1-\epsilon)},$$

where $\eta = \frac{1}{\sqrt{s}} \Gamma(1 + \sqrt{2\log(\frac{1}{\delta})})$, $C = \psi - \frac{(M - \kappa(1-\epsilon)^2)}{2K\kappa^2(1-\epsilon)^2}$. *where* t_0 *is the last iteration where the models were synced,* $\psi = \frac{\alpha^\star \beta}{M(1+\epsilon)}$, *and* $C = \frac{\psi(1-\epsilon)^3}{2}$.

Proof. The proof is presented in Appendix 3. □

Remark 4. *The theorem shows that LocalNewton with high probability produces a descent direction, provided that the error floor is sufficiently small, i.e. for sufficiently large* s *(since* η *is proportional to* $1/\sqrt{s}$*). Observe that the convergence rate here is no longer linear. In other words, we are trading-off the rate of convergence for local iterations* $(L > 1)$.

Remark 5. *Choosing* $\delta = 1/\mathsf{poly}(K, L)$, *we get that the theorem holds with probability at least* $1 - 1/\mathsf{poly}(K, L)$. *Note that this is not restrictive since the dependence on* δ *is logarithmic.*

While the theoretical guarantees for $L > 1$ in Theorem 4.4 are not as strong as those for $L = 1$ in Theorem 4.3 (linear versus sublinear convergence), empirically we observe a fast rate of convergence even when $L > 1$ (see Section 5 and Appendix 5 for empirical results). Nevertheless, to the best of our knowledge, Theorem 4.4 is the first to show a descent guarantee for a distributed second-order method without synchronizing at every iteration. Obtaining a better rate of convergence for general L, with or without error floor, is an interesting and relevant future research direction.

5 EMPIRICAL EVALUATION

In this section, we present an empirical evaluation of our approach when solving a large-scale logistic regression problem. We ran our experiments on AWS Lambda workers using the PyWren [Jonas et al., 2017] framework. AWS Lambda is a *serverless* computing platform which uses a high-latency cloud storage (AWS S3) to exchange data with the workers. The master is a *serverful* AWS EC2 machine of type `m4.4xlarge` which co-ordinates with the serverless workers and holds the central model during the entire training run. We ran experiments on the real-world datasets described in Table 1 (obtained from LIBSVM [Chang and Lin, 2011]).

We compare the following distributed optimization schemes for the above datasets: 1. Local SGD [Stich, 2018]: The workers communicate their models once every epoch, where training on one epoch implies applying SGD (with mini-batch size one) over one pass of the dataset stored locally at the worker. The best step-size was obtained through hyper-parameter tuning.

Dataset	Training samples (n)	Features (d)	Testing samples
w8a	$48,000$	300	$15,000$
Covtype	$500,000$	2916	$81,000$
EPSILON	$400,000$	2000	$100,000$
a9a	$32,000$	123	$16,000$
ijcnn1	$49,000$	22	$91,000$

Table 1: Datasets considered for experiments in this paper

2. BFGS [Fletcher, 2013]: BFGS is a popular quasi-Newton method that estimates an approximate Hessian from the gradient information from previous iterations. The step-size was obtained using backtracking line-search. The best step-size was obtained through hyperparameter tuning.

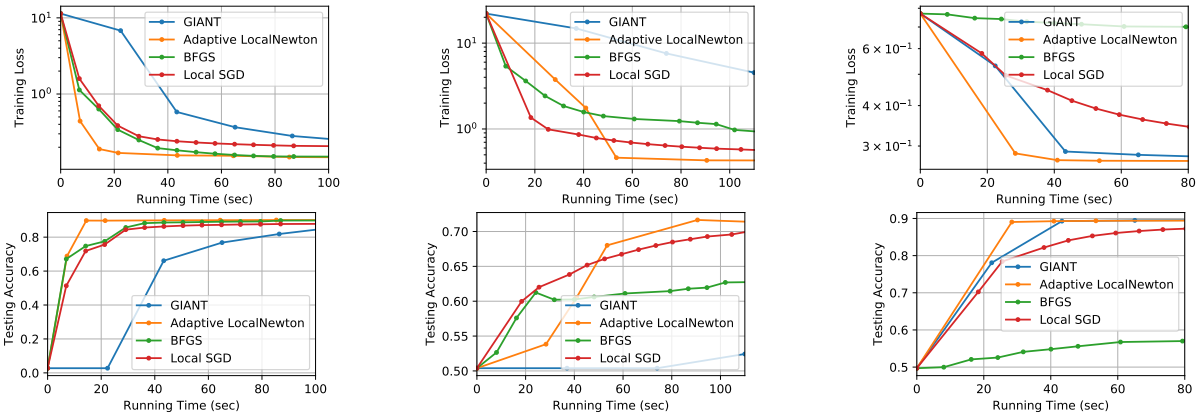
3. GIANT [Wang et al., 2018]: A state-of-the-art distributed second order algorithm proposed in Wang et al. [2018]. The authors show that GIANT outperforms many popular schemes such as DANE, AGD, etc. The step-size was obtained using distributed line-search as described in Wang et al. [2018].

4. Adaptive LocalNewton: For all the considered datasets, Adaptive LocalNewton gradually reduces L if the loss function stops decreasing, starting from $L = 3$ in the first round of communication. In general, during the later stages of optimization, it switches to GIANT owing to its convergence to the optimal solution when $\bar{\mathbf{w}}_t$ is sufficiently close to $\mathbf{w}^*$. The step-size was obtained using backtracking line-search locally at each worker as described in Algorithm 1.

We also implemented the distributed second-order optimization scheme from Duenner et al. [2018] with moderate hyperparameter tuning, and observed that its performance is either comparable or worse than the baseline GIANT Wang et al. [2018]. Hence, for clarity, we omit those results from our plots.

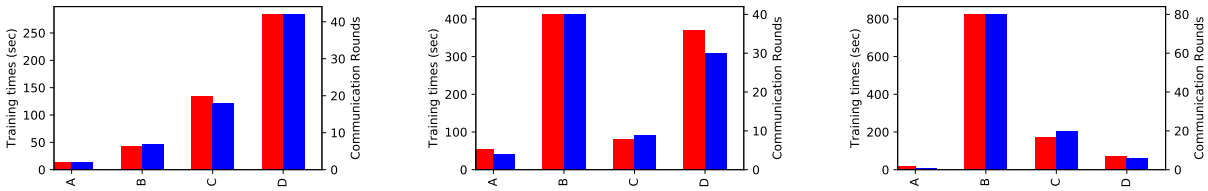
For all the experiments presented in this paper, we fixed the number of workers, K, to be 100. Hence, the number of samples per worker, $s = n/100$, for all datasets. The regularization parameter was chosen to be $\gamma = 1/n$. Note that there are several other schemes–such as AGD [Nesterov, 2014], DANE [Shamir et al., 2014] and SVRG Johnson and Zhang [2013]–that have been proposed in the literature for communication-efficient optimization. However, most of these schemes have been shown to be outperformed by one of Local SGD, BFGS or GIANT, and hence, we do not perform the comparison again. In Fig. 3, we plot the training loss and testing accuracy for w8a, covtype[7] and EPSILON

[7]The covtype dataset has $d = 54$ features and it does not



(a) w8a dataset (b) Covtype dataset (c) EPSILON dataset

Figure 3: Experiments on the w8a, Covtype and EPSILON datasets on AWS Lambda. Both in terms of training loss and testing accuracy, Adaptive LocalNewton converges to the optimal value at least 50% faster than existing schemes.



(a) w8a dataset (b) Covtype dataset (c) EPSILON dataset

Figure 4: Training times (red bars) and communication rounds (blue bars) required to reach the same training loss of $0.19, 0.65$ and 0.3 for the w8a, Covtype and EPSILON datasets, respectively, on AWS Lambda. Here, A: Adaptive LocalNewton, B: BFGS, C: Local SGD, D: GIANT.

datasets (see Fig. 1 in Appendix 5 for experiments on a9a and ijcnn1 datasets). For all the datasets considered, Adaptive LocalNewton significantly outperforms its competitors in terms of time required to reach the same training loss (or testing accuracy). Furthermore, since the number of workers, K, is fixed and serverless platform charges are proportional to the total CPU hours, end-to-end training time is directly proportional to the costs charged by AWS for training.

In Fig. 4, we highlight the fact that runtime savings on AWS Lambda are a direct consequence of significantly fewer rounds of communication. Specifically, to reach the same training loss, we plot the training times and communication rounds as bar plots for three datasets, and we note that savings in communication rounds results in commensurate savings on end-to-end runtimes on AWS Lambda. In Fig. 2 in Appendix 5, we provide detailed plots for training losses and testing accuracies with respect to communication rounds for all the five datasets.

perform well with logistic regression. Hence, we apply polynomial feature extension (using pairwise products) to increase the number of features to $d^2 = 2916$.

6 CONCLUSION

The practicality of second-order optimization methods has been questioned since naive ways to implement them require large compute and power storage to work with the Hessian. However, in the last few decades, trends such as Moore's law have made computation faster and memory cheaper, while improvements in communication costs have been at best marginal. These trends, combined with a flurry of efficient but approximate algorithms [Pilanci and Wainwright, 2017, Roosta-Khorasani and Mahoney, 2016a,b], have revived interest in second-order methods. In this paper, we identify and concretize the role that second-order methods—combined with local optimization algorithms—can play in reducing the communication costs during distributed training, in particular in serverless environments. Since second-order information has recently been used to develop state-of-the-art methods for deep neural networks with extremely large model sizes [Dong et al., 2019, Yao et al., 2019, Shen et al., 2020, Yao et al., 2020], we expect that methods such as ours will play a significant role in motivating and designing next-generation communication-efficient algorithms for fast distributed training of machine learning models.

References

Jayadev Acharya, Chris De Sa, Dylan Foster, and Karthik Sridharan. Distributed learning with sublinear communication. In *International Conference on Machine Learning*, pages 40–50, 2019.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *International Conference on Machine Learning*, pages 560–569, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

James Demmel. Communication-avoiding algorithms for linear algebra and beyond. In *2013 IEEE 27th Int. Sym. on Parallel and Distributed Processing*, pages 585–585, May 2013. doi: 10.1109/IPDPS.2013.123.

Michał Dereziński and Michael W Mahoney. Distributed estimation of the inverse hessian by determinantal averaging. In *Advances in Neural Information Processing Systems 32*, pages 11405–11415. Curran Associates, Inc., 2019.

Michał Dereziński, Burak Bartan, Mert Pilanci, and Michael W Mahoney. Debiasing distributed second order optimization with surrogate sketching and scaled regularization. In *Advances in Neural Information Processing Systems*, volume 33, pages 6684–6695, 2020.

Aymeric Dieuleveut and Kumar Kshitij Patel. Communication trade-offs for local-sgd with large step size. In *Advances in Neural Information Processing Systems*, pages 13579–13590, 2019.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 293–302, 2019.

Celestine Duenner, Aurelien Lucchi, Matilde Gargiani, An Bian, Thomas Hofmann, and Martin Jaggi. A distributed second-order algorithm you can trust. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1358–1366. PMLR, 10–15 Jul 2018.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

Avishek Ghosh, Raj Kumar Maity, and Arya Mazumdar. Distributed newton can communicate less and resist byzantine workers. *arXiv preprint arXiv:2006.08737*, 2020a.

Avishek Ghosh, Raj Kumar Maity, Arya Mazumdar, and Kannan Ramchandran. Communication efficient distributed approximate newton method. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2539–2544. IEEE, 2020b.

Ghosh, Avishek, Maity, Rajkumar, Kadhe, Swanand, Mazumdar, Arya, and Ramachandran, Kannan. Communication efficient and byzantine tolerant distributed learning. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2545–2550, 2020.

Vipul Gupta, Dominic Carrano, Yaoqing Yang, Vaishaal Shankar, Thomas Courtade, and Kannan Ramchandran. Serverless straggler mitigation using local error-correcting codes. *IEEE International Conference on Distributed Computing Systems*, 2020a.

Vipul Gupta, Santiago Akle Serrano, and Dennis DeCoste. Stochastic weight averaging in parallel: Large-batch training that generalizes well. In *International Conference on Learning Representations*, 2020b.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, pages 11080–11092, 2019.

Joseph M Hellerstein, Jose Faleiro, Joseph E Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. Serverless computing: One step forward, two steps back. *arXiv preprint arXiv:1812.03651*, 2018.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems*, pages 13144–13154, 2019.

Ali Jadbabaie, Asuman Ozdaglar, and Michael Zargham. A distributed newton method for network optimization. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2736–2741. IEEE, 2009.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. Occupy the cloud: distributed computing for the 99%. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 445–451. ACM, 2017.

Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019.

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

Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't Use Large Mini-batches, Use Local SGD. In *International Conference on Learning Representations*, 2020.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Prathamesh Mayekar and Himanshu Tyagi. Limits on Gradient Compression for Stochastic Optimization. *arXiv e-prints*, art. arXiv:2001.09032, jan 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282, 2017.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition, 2014. ISBN 1461346916, 9781461346913.

Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Jour. on Opt.*, 27:205–245, 2017.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Pöczos, and Alex Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2647–2655, 2015.

Farbod Roosta-Khorasani and Michael W. Mahoney. Sub-Sampled Newton Methods I: Globally Convergent Algorithms. *arXiv e-prints*, art. arXiv:1601.04737, January 2016a.

Farbod Roosta-Khorasani and Michael W. Mahoney. Sub-Sampled Newton Methods II: Local Convergence Rates. *arXiv e-prints*, art. arXiv:1601.04738, January 2016b.

Ohad Shamir, Nathan Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, 2014.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020.

Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takác, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

Sebastian U Stich. Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458, 2018.

Rasul Tutunov, Haitham Bou-Ammar, and Ali Jadbabaie. Distributed newton method for large-scale consensus optimization. *IEEE Transactions on Automatic Control*, 64 (10):3983–3994, 2019.

Shusen Wang, Farbod Roosta-Khorasani, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. In *Advances in Neural Information Processing Systems 31*, pages 2332–2342. Curran Associates, Inc., 2018.

Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184 (1):35–70, 2020.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. PyHessian: Neural Networks Through the Lens of the Hessian. *arXiv preprint arXiv:1912.07145*, 2019.

Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*, 2020.

Yuchen Zhang and Xiao Lin. DiSCO: Distributed Optimization for Self-Concordant Empirical Loss. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 362–370, Lille, France, 07–09 Jul 2015. PMLR.