2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) | 978-1-6654-8053-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/PIMRC54779.2022.9977466

On Analog Distributed Approximate Newton with Determinantal Averaging

Ganesh Sharma
Hamilton Institute
Maynooth University
Maynooth, Ireland
ganesh.sharma.2019@mumail.ie

Subhrakanti Dey
Department of Electrical Engineering
Uppsala University
Uppsala, Sweden
subhrakanti.dey@angstrom.uu.se

***Abstract*—This paper considers the problem of communication and computation-efficient distributed learning via a wireless fading Multiple Access Channel (MAC). The distributed learning task is performed over a large network of nodes containing local data with the help of an edge server coordinating between the nodes. The information from each distributed node is transmitted as an analog signal through a noisy fading wireless MAC, using a common shaping waveform. The edge server receives a superposition of the analog signals, computes a new parameter estimate and communicates it back to the nodes, a process which continues until an appropriate convergence criterion is met. Unlike typical Federated learning approaches based on communication of local gradients and averaging at the edge server, in this paper, we investigate a scenario where the local nodes implement a second order optimization technique known as *Determinantal Averaging*. The communication complexity at each iteration per node of this method is the same as any gradient based method, i.e. $O(d)$, where d is the number of parameters. To reduce the computational load at each node, we also employ an approximate Newton method to compute the local Hessians. Under the usual assumptions of convexity and double differentiability on the local objective functions, we propose an algorithm titled Distributed Approximate Newton with Determinantal Averaging (DANDA). The state-of-art first and second-order distributed optimization algorithms are numerically compared with DANDA on a standard dataset with least squares based local objective functions (linear regression). Simulation results illustrate that DANDA not only displays faster convergence compared to gradient-based methods, but also compares favourably with exact distributed Newton methods, such as LocalNewton.**

***Index Terms*—Distributed Learning (DL), Analog Transmission, Fading Multiple Access Channel (MAC), Approximate Newton methods**

I. Introduction

Privacy concerns often restrict the application of Machine learning (ML). Conventionally, ML tasks are performed in a server containing data gathered from multiple edge devices or nodes (e.g. mobiles, sensors, etc.). It is difficult to preserve privacy when the user data have to be collected from the nodes for model training in "centralized" learning tasks. Distributed Machine Learning (DML) provides a method to train the ML model locally, without collecting data from the nodes. An edge server is often used in DML that coordinates between the nodes. While training, the nodes share training parameters to the edge server which aggregates and provides a new parameter estimate back to the nodes. This process continues until the training parameters attain a global optimum within a certain tolerance. This way of learning discards the movement of data for model training from the end-user, thereby securing privacy as well as reducing communication overhead.

A. Related Work

Our work is related to the field of Federated Learning (FL) [1] where the nodes contribute in the learning process with the help of an edge server. The edge devices exchange local gradient information based on local data with the edge server and perform the learning task using local stochastic gradient descent (SGD) techniques. The conventional FL algorithms use orthogonal access channel for communicating information from the nodes, which exhausts the bandwidth as the number of nodes increases. Numerous studies show the application of distributed-SGD over wireless Multiple Access Channel (MAC) using analog uncoded transmission, utilizing its *over-the-air* additive nature of the wireless channel, and reducing the channel bandwidth requirement e.g. [2]–[7]. The works of [2]–[4] use analog MAC transmission method with scheduling schemes and power control to compensate for the wireless fading channels. The GBMA algorithm in [5] works directly with distorted gradients transmitted over MAC using analog transmission, while the authors in [6] propose the optimization of the parameters in the transceiver with consideration of the "non-stationarity" in the local gradients based on a simple feedback variable. In [7], the authors propose pre-coding at the nodes and scaling at the parameter server to mitigate the effect of channel noise.

The iterative algorithms based on transmission of local gradients have a significant communication overhead due of their slow convergence (i.e. a large number of communication rounds). To alleviate the communication cost, second order optimization based methods can be used at the local nodes, although they are computationally expensive. Extensive literature on reducing the communication overhead by using quasi-Newton methods is present. These include DANE [8], that approximates the Newton step in a distributed manner using

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049

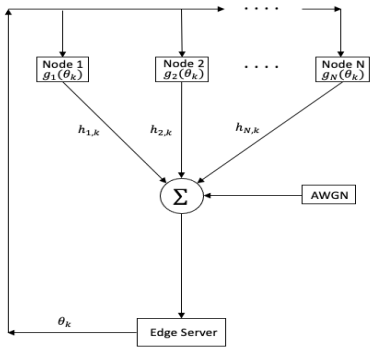


Fig. 1: Transmission Scheme over a wireless MAC

Bregman divergence, DONE [9], that produces an approximate newton direction using classical Richardson iteration on each edge node, whereas GIANT [10] and DISCO [11] approximate the Hessian using conjugate gradient methods. One method of reducing the communication overhead is using the Newton method locally and instead of sharing newton direction for global parameter vector update, optimize the parameter vector locally and share the updated parameters as done in the algorithm LocalNewton [12]. The authors in [13] compute Newton directions locally and share the local optimum parameter vector to the edge server by adopting a nonconvex low-rank beamforming approach with over-the-air computation via difference-of-convex-functions (DC) programming. Most second order Newton type methods, however require computation of the inverse of the Hessians locally, which is computationally expensive, thus warranting various forms of approximate Newton methods. In a recent work [14], the authors determine the inverse Hessian times the gradient product vector using the Alternating Direction Method of Multipliers (ADDM) in an effort to reduce computation costs.

B. Our Contributions

In this paper, we propose a novel distributed learning method based on a computationally efficient approximate Newton method based on the technique of determinantal averaging [15] at each node. The nodes communicate training parameters with the edge server using analog uncoded transmission over a fading wireless MAC, with a communication overhead of $O(d)$ at each iteration, where d is the number of model parameters to be optimized. This distributed learning algorithm, titled DANDA, is empirically tested in a linear regression task with the Million Song Dataset [17], in a simulated communication setting with its performance averaged over multiple random channel realizations. Comprehensive numerical results illustrate significantly faster convergence of DANDA compared to first order methods such as GBMA, and comparable energy consumption and computational complexity with state of the art second order methods.

The rest of the paper is organized as follows. Section 2 provides the distributed learning and the communication models. Section 3 provides the theoretical background of the algorithm DANDA, followed by a description of its implementation. Section 4 provides a comprehensive set of experimental results on its comparative performance with existing algorithms, followed by some concluding remarks in Section 5.

II. SYSTEM MODEL

A. Distributed Learning Model

We consider a distributed wireless edge learning problem with N nodes and an edge server for aggregation of information collected from the nodes over a fading wireless MAC. The global objective function, $F(\boldsymbol{\theta})$, is an average of the local objective functions, $f_n(\boldsymbol{\theta})$, as described in the following optimization problem:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\Big\{F(\boldsymbol{\theta})=\frac{1}{N}\sum_{n=1}^{N}f_n(\boldsymbol{\theta})\Big\},\ f_n(\boldsymbol{\theta})=\frac{1}{b_n}\sum_{i=1}^{b_n}\ell_n(\boldsymbol{\theta}^T\boldsymbol{x}_{i,n}) \tag{1}$$

where $\boldsymbol{\theta}$ is a $d\times 1$ parameter vector over which the optimization of the global objective function takes place, ℓ_n is the loss function at the n-th node, $\boldsymbol{x}_{i,n}$ is the i^{th} input row for n^{th} node, and b_n is the size of dataset in the n^{th} node. As shown in Fig. 1, the edge server broadcasts an initial parameter vector, $\boldsymbol{\theta}_0$ to all the nodes, to which each node determines a function of the local information on its data. These functions are then communicated to the edge server over the wireless MAC as analog signals with a common shaping waveform, and due to the superposition property of the wireless medium, get aggregated over the transmission channel, and consequently provides the global descent direction at the edge server.

B. Communication Model

We consider a wireless MAC, where each edge node transmits information as a common analog waveform and the superposition of the N transmitted waveforms is received at the edge server as shown in Fig. 1. In the case where the number of features, d, is much smaller than the number of nodes, N, this communication scheme solves the issue of high bandwidth requirement in distributed ML. We assume time is slotted, with slot duration T, with the k-th slot being $t_k \le t \le t_k + T$. Let $\boldsymbol{s}(t) = (s_1(t), s_2(t), ...s_d(t)), 0 < t < T$, be a vector of d orthogonal base-band equivalent normalized waveforms, satisfying $\int_0^T s_m^2(t)dt = 1, \int_0^T s_m(t)s_t(t)dt = 0$,for $m \neq n$. Each node n experiences at the k-th slot a block fading channel $\tilde{h}_{n,k}$ with gain $h_{n,k} \triangleq |\tilde{h}_{n,k}| \in \mathbb{R}_+$ and phase $\phi_{n,k} \triangleq \angle\tilde{h}_{n,k} \in \{-\pi, \pi\}$. The channel fading is assumed independent and identically distributed (*i.i.d.* across the nodes), and time slots, with mean μ_h and variance σ_h^2. We also assume that each iteration of distributed learning algorithm is aligned with the time-slots. In other words, the fading channel coefficients are constant within a single iteration, but change from iteration to iteration in a statistically independent manner.

NOTE: The phase lag due to channel experienced by each node $\phi_{n,k}$ is assumed to be known at each node and therefore cancelled before transmitting a signal. This is the underlying principle of transmission over a coherent MAC, which obviously requires distributed transmit beamforming, and can

be difficult to achieve across a large number of nodes. Note however, that even if the channel phases are not accurately accounted for, and there is some residual phase offset, the individual signals arriving at the edge server will still have positive coefficients as long as the phase offsets are small [5]. A description of useful notations is provided in Table I.

III. Distributed Newton algorithms with Determinantal Averaging

We aim to solve the empirical risk minimization problem described in (1) where $f_n(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}$, for all $n \in [N] = \{1, 2...N\}$. The function $f_n(\cdot)$ satisfies the following assumptions:

Assumption 1: $f_n(\cdot)$ is *Convex and twice differentiable.*

Assumption 2: $f_n(\cdot)$ has a *Lipschitz continuous Hessian: For any* $n \in [N]$*, there exists a constant L, such that* $\forall \boldsymbol{\theta}, \boldsymbol{\theta'} \in \mathbb{R}^d$,

$$||\boldsymbol{H}_n(\boldsymbol{\theta}) - \boldsymbol{H}_n(\boldsymbol{\theta'})||_2 \leq L||\boldsymbol{\theta} - \boldsymbol{\theta'}||$$

where, $\boldsymbol{H}_n(\boldsymbol{\theta}) = \nabla^2 f_n(\boldsymbol{\theta})$

Note: The assumptions mentioned above are standard in the machine learning literature and are satisfied by several standard cost functions, including the linear regression based least squares cost function.

A. Distributed Newton with Determinantal Averaging: DNDA

In the distributed setting, we have N nodes and each node carries a subset $\mathcal{S}_n \subset [\mathcal{N}]$ of size b_n (i.e. $b_n = |\mathcal{S}_n|$), for all $n \in [N] = \{1, 2...N\}$, where $\mathcal{N}$ is the total number of data points in the original dataset. The samples $\mathcal{S}_n$ at each node are obtained randomly without replacement, hence $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \forall i, j \in [N], i \neq j$.

Note that a typical second order optimization algorithm based on approximating the global Newton step, $\boldsymbol{H}^{-1}(\boldsymbol{\theta})\boldsymbol{G}(\boldsymbol{\theta})$ for $F(\boldsymbol{\theta})$, involves aggregating the local Hessians and the local gradients from the local Newton steps, $\boldsymbol{H}_n^{-1}(\boldsymbol{\theta})\boldsymbol{g}_n(\boldsymbol{\theta})$ for $f_n(\boldsymbol{\theta})$, from the nodes. However, obtaining the Hessian for the global Newton's step by naive aggregation of the local Newton steps suffers an inversion bias as $\mathbb{E}(\boldsymbol{H}_n^{-1}(\boldsymbol{\theta}_k)) \neq \boldsymbol{H}^{-1}(\boldsymbol{\theta}_k)$. Derezinski *et. al.* in [15] proved that a weighted sum of local inverse hessian times the global gradient, asymptotically converges to global inverse hessian times the global gradient. They call this Determinantal Averaging, as given below. Define

$$\widehat{\boldsymbol{H}^{-1}(\boldsymbol{\theta})\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{\sum_{n=1}^{N} a_n \boldsymbol{H}_n^{-1}(\boldsymbol{\theta})\boldsymbol{G}(\boldsymbol{\theta})}{\sum_{n=1}^{N} a_n} \tag{2}$$

where $a_n = \det(\boldsymbol{H}_n)$ and $\boldsymbol{G}(\boldsymbol{\theta})$ is the global gradient. As $N \to \infty$, $\widehat{\boldsymbol{H}^{-1}(\boldsymbol{\theta})\boldsymbol{G}}(\boldsymbol{\theta}) \to \boldsymbol{H}^{-1}(\boldsymbol{\theta})\boldsymbol{G}(\boldsymbol{\theta})$ almost surely. Note that this estimation requires the product of local inverse Hessian and the global gradient[1] which is a vector of size $d \times 1$. As the number of nodes, $N \to \infty$, the estimate of inverse of the global Hessian $\hat{\boldsymbol{H}}^{-1} \to \boldsymbol{H}^{-1}$. The convergence guarantee offered by this method is given by:

[1]However, as shown in [14], the local Newton's step can be replaced by the product of local hessian and local gradient, $(\boldsymbol{H_n^{-1}(\theta)g_n(\theta)})$, as long as $g_n(\theta))$ is an i.i.d. unbiased estimate of the global gradient.

Corollary 6* [15]: For any* $\delta, \eta \in (0, 1)$ *if the expected local sample size satisfies* $K \geq C\eta^{-2}\mu d^2 log^3 \frac{d}{\delta}$ *then under Assumption 2*

$$||\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta^*}|| \leq max\{\frac{\eta}{\sqrt{N}}\sqrt{K}||\boldsymbol{\theta} - \boldsymbol{\theta^*}||, \frac{2L}{\lambda_{min}}||\boldsymbol{\theta} - \boldsymbol{\theta^*}||^2\}$$

where $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \widehat{\boldsymbol{H}^{-1}(\boldsymbol{\theta})\boldsymbol{G}}(\boldsymbol{\theta})$,

holds with probability at least $1 - \delta$*, where C is an absolute constant,* $\mu = \frac{1}{d}\max_i \ell''(\boldsymbol{\theta}^T \boldsymbol{x}_i)(\boldsymbol{x}_i{}^T \nabla^{-2} F(\boldsymbol{\theta})\boldsymbol{x}_i)$*, K and* λ_{min} *are the condition number and smallest eigenvalues of* $\nabla^2 F(\theta)$.

It can be observed that as $N \to \infty$ the above bound exhibits quadratic convergence similar to the exact Newton's method.

In our analog transmission based distributed learning algorithm, the signal to be transmitted by the n^{th} node at the k^{th} iteration is sent to the edge server in two halves of the time slot, representing functions of the numerator and the denominator in (2). The signals transmitted from the nodes in the first ($t_k \leq t < t_k + \frac{T}{2}$) and second half ($t_k + \frac{T}{2} \leq t < t_k + T$) of the k-th time slot, respectively, as

$$y_n^1(\boldsymbol{\theta}_k, t) \triangleq \sqrt{\alpha P} e^{-j\phi_{n,k}} a_n (\boldsymbol{H}_n^{-1}(\boldsymbol{\theta}_k)\boldsymbol{g}_n(\boldsymbol{\theta}_k))^T \boldsymbol{s}(t), \tag{3}$$

$$y_n^2(\boldsymbol{\theta}_k, t) \triangleq \sqrt{(1-\alpha)P} e^{-j\phi_{n,k}} a_n s'(t) \tag{4}$$

Here, $a_n = \det(\boldsymbol{H}_n(\boldsymbol{\theta}))$, $e^{-j\phi_{n,k}}$ is the phase correction factor to produce a positive channel gain at the receiver, $\boldsymbol{s}(t)$ is the vector of orthogonal normalised waveforms to carry the numerator part of (2) and $s'(t)$ is a scalar normalised waveform function to carry the denominator part of (2). P is a node power adjustment factor (PAF), which is distributed among the two signals such that αP is the power to the first signal representing the numerator and $(1-\alpha)P$ is the power to the denominator, where $\alpha \in (0, 1)$. Signals received at the edge server are superimposed on each other due to the analog nature of the transmitted signals. The two received signals in the first and second halves of the k-the time slot are, respectively:

$$\boldsymbol{r}_k^1(t) = \sum_{n=1}^{N} \sqrt{\alpha P} h_{n,k} a_n (\boldsymbol{H}_n^{-1}(\boldsymbol{\theta}_k)\boldsymbol{g}_n(\boldsymbol{\theta}_k))^T \boldsymbol{s(t)} + w_k^1(t), \tag{5}$$

$$r_k^2(t) = \sum_{n=1}^{N} \sqrt{(1-\alpha)P} h_{n,k} a_n s'(t) + w_k^2(t) \tag{6}$$

Here $\boldsymbol{w}_k^1$ and w_k^2 are additive independent white Gaussian noise (AWGN) processes distributed as $\mathcal{N}(0, \sigma_w^2 \boldsymbol{I_d})$ and $\mathcal{N}(0, \sigma_w^2)$ respectively. After matched filtering and some readjustments, the demodulated signals at the edge server are:

$$\boldsymbol{v}_k^1 = \frac{1}{N}\sum_{n=1}^{N} h_{n,k} a_n \boldsymbol{H}_n^{-1}(\boldsymbol{\theta}_k)\boldsymbol{g}_n(\boldsymbol{\theta}_k) + \boldsymbol{w}_k^1 \tag{7}$$

$$v_k^2 = \frac{1}{N}\sum_{n=1}^{N} h_{n,k} a_n + w_k^2 \tag{8}$$

where $\boldsymbol{w}_k^1 \sim \mathcal{N}(0, \frac{\sigma_w^2 \boldsymbol{I_d}}{N^2 \alpha P})$ and $w_k^2 \sim \mathcal{N}(0, \frac{\sigma_w^2}{N^2(1-\alpha)P})$. The edge server receives the superposition of the respective signals

Symbol	Description	Symbol	Description	Symbol	Description
N	number of nodes	$\boldsymbol{H}_n$	local Hessian for n^{th} node	a_n	determinant of local Hessian at n^{th} node
n	node index	$\boldsymbol{G}$	global gradient	α	power distribution factor (PDF), $0 < \alpha < 1$
d	number of features	$\boldsymbol{g}_n$	local gradient for n^{th} node	b_n	size of dataset at n^{th} node, $\lvert\boldsymbol{\mathcal{S}}_n\rvert$
R	entire dataset	$\boldsymbol{s}(t)$	d orthogonal normalized waveforms	$h_{n,k}$	channel gain for n^{th} node and k^{th} iteration
k	iteration index	$\boldsymbol{\mathcal{S}}_n$	dataset at n^{th} node	$\boldsymbol{w}_k$	thermal noise at k^{th} iteration, $\boldsymbol{w}_k \sim \mathcal{N}(0, \sigma_w^2 \boldsymbol{I}_d)$
$\boldsymbol{\theta}$	parameter vector	S_n	sub-sampled dataset at n^{th} node	β	generic learning rate
$\boldsymbol{\theta^*}$	optimum parameter vector	$\boldsymbol{H}$	global Hessian	P	power adjustment factor (PAF)

TABLE I: Symbols and their description

from all the nodes with channel distortion and AWGN. The ratio of two signals, $\frac{\boldsymbol{v}_k^1}{\boldsymbol{v}_k^2}$, gives noisy and distorted version of the Newton's step, that is used for parameter updates as follows,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta \frac{\boldsymbol{v}_k^1}{\boldsymbol{v}_k^2} \tag{9}$$

where β is an appropriately scaled learning rate.

Note that the communication overhead per node involves a $d+1$ dimensional vector at each iteration, requiring a similar communication overhead compared to gradient based methods. The difficulty of course lies in computing the inverse local Hessian at the nodes, which in general, costs $O(d^3)$ computations per iteration. In Section III-C, we will address this issue by using a quasi-Newton method to approximate the local Hessians.

B. DNDA with Recursive Least Squares

Consider the special case where the local cost functions are regularized least square loss functions (i.e. linear regression), given as follows:

$$f_n(\boldsymbol{\theta}) = \ell_n(\boldsymbol{\theta}^T \boldsymbol{x}_n) = \frac{1}{2}||\boldsymbol{\theta}^T \boldsymbol{x}_n - \boldsymbol{y}_n||_2^2 + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 \tag{10}$$

It is easy to check that in this case the local Hessians are independent of the parameter vector $\boldsymbol{\theta}$ and hence is constant over the iterations of DNDA algorithm. Therefore the denominator of the (2), which involves the sum of the determinants of the local Hessians, can be estimated using a recursive least squares estimation method. For the k^{th} iteration, the received denominator of (2) is a distorted version of the determinant of the local hessian, a_n summed over all the nodes, given by v_k^2 in (8).

By observing the received signal v_k^2 over $\bar{T}$ iterations, We can estimate the local hessian determinants, a_n at k^{th} iteration using an ordinary least square problem as follows:

$$\hat{\boldsymbol{a}}_k = (\sum_{k=1}^{\bar{T}} \boldsymbol{h}_k \boldsymbol{h}_k^T)^{-1} (\sum_{k=1}^{\bar{T}} \boldsymbol{h}_k z_k)$$

where $\boldsymbol{h}_k$ is a vector of channel gains observed by the nodes 1 to N for k^{th} iteration, $\hat{\boldsymbol{a}}_k$ is a vector of estimated local hessian determinants for nodes 1 to N for k^{th} iteration and $z_k = N v_k^2$. A recursive least squares (RLS) method can be used to estimate $\boldsymbol{a}_n$ iteratively as follows,

$$\begin{aligned} \hat{\boldsymbol{a}}_k &= \hat{\boldsymbol{a}}_{k-1} + \mathcal{K}_k (z_k - (\boldsymbol{h}_k)^T \hat{\boldsymbol{a}}_{k-1}) \\ \mathcal{K}_k &= \mathcal{P}_k \boldsymbol{h}_k, \; \mathcal{P}_k = \frac{\mathcal{P}_{k-1}}{\boldsymbol{h}_k^T \mathcal{P}_{k-1} \boldsymbol{h}_k + 1} \end{aligned} \tag{11}$$

where, $\mathcal{K}_k$ is the gain vector for the k^{th} iteration and $\mathcal{P}_k$ represents $(\sum_{k=1}^{\bar{T}} \boldsymbol{h}_k \boldsymbol{h}_k^T)^{-1}$, computed in a recursive way using the matrix inversion lemma.

This method avoids the direct use of a distorted sum of local hessian determinants, $\sum_n a_n$, and instead uses an estimate of it. Using a recursive least squares estimate of $\sum_n a_n$ provides better asymptotic convergence compared to the Distributed Newton with Determinantal Averaging (DNDA) algorithm using the distorted sum of local Hessians, as shown in the Experimental Results section.

C. Distributed Approximate Newton with Determinantal Averaging: DANDA

Algorithm 1: DANDA

input: Local function $f_n(\cdot)$ at the n^{th} node; Initial iterate $\boldsymbol{\theta_0} \in \mathbb{R}^d$; Regularization constant (λ); learning rate (β); PAF (P); Optimal PDF (α^*); Sub-sample sizes (S_n^k); Convergence Parameter (ϵ)

define: $[\boldsymbol{U}_r, \boldsymbol{\Lambda}_r]$ =TruncatedSVD$_r(\boldsymbol{H}_{S_n})$ as rank-r truncated SVD of local sub-sampled hessian ($\boldsymbol{H}_{S_n}$) with $(\boldsymbol{\Lambda}_r)_{ii} = \lambda_i$;
$k = 1$; $\boldsymbol{\theta}_k = \boldsymbol{\theta}_0$; $F(\boldsymbol{\theta}_k) = \frac{1}{N} \sum_{i=1}^n f_n(\boldsymbol{\theta}_k)$

while $\frac{|F(\boldsymbol{\theta_k}) - F(\boldsymbol{\theta_{k+1}})|}{F(\boldsymbol{\theta_k})} \geq \epsilon$ **do**
- Local training process;
- **for** $n = 1$ *to* N **do**
  - Compute local gradient
  - $\boldsymbol{g}_n = \nabla f_n(\boldsymbol{\theta}_k)$
  - Compute local hessian using sub-samples
  - Let $\boldsymbol{H}_{S_n^k} = \frac{1}{|S_n^k|} \sum_{i \in S_n^k} \ell_n''(\boldsymbol{\theta}_k^T \boldsymbol{x}_{i,n})$
  - $\hat{\boldsymbol{H}}_{S_n^k}^{-1} = \lambda_{r+1}^{k-1} \boldsymbol{I}_d + \boldsymbol{U}_r^k (\Lambda_r^{k^{-1}} - \lambda_{r+1}^{k-1} \boldsymbol{I}_r) \boldsymbol{U}_r^{k^T}$
  - Compute the numerator $\boldsymbol{N}_n^k$ and the denominator $\boldsymbol{D}_n^k$ for the determinantal averaging
  - $a_n^k = \det(\hat{\boldsymbol{H}}_{S_n^k})$
  - $\boldsymbol{N}_n^k = a_n \hat{\boldsymbol{H}}_{S_n^k}^{-1} \boldsymbol{g}_n$
  - $\boldsymbol{D}_n^k = a_n$
- Aggregate distorted $\hat{\boldsymbol{N}}_k$ and $\hat{\boldsymbol{D}}_k$
- $\hat{\boldsymbol{N}}_k = \frac{1}{N} \sum_{n=1}^N h_n^k N_n^k + \boldsymbol{w}_1^k$
- $\hat{\boldsymbol{D}}_k = \frac{1}{N} \sum_{n=1}^N h_n^k D_n^k + \boldsymbol{w}_2^k$
- Update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta \frac{\hat{\boldsymbol{N}}_k}{\hat{\boldsymbol{D}}_k}$
- $k \leftarrow k+1$

output: $\boldsymbol{\theta}_k$

Computing the inverse Hessian in Newton's step is expensive, especially for high-dimensional datasets, and computational constraints at the edge nodes may render it even prohibitive. Here we adopt an approximate Newton's step using sub-sampled Hessians, as presented by Erdogdu and Monta-

nari in their algorithm called NewSamp [16]. This algorithm has been shown to significantly reduce computational cost, while still preserving the fast convergence rate of a second-order method. Other than convexity and differentiability of the local cost functions and Lipschitz continuity of the Hessians mentioned above, in this section, we need two additional assumptions:

Assumption 3: Bounded Hessian: $\forall i = 1, 2, .., n$, *the Hessian of the function* $f_i(\theta), \nabla^2_\theta f_i(\theta)$, *is upper bounded by an absolute constant K, i.e.,* $\max_{i \leq n} ||\nabla^2_\theta f_i(\theta)|| \leq K$.

Assumption 4: Lipschitz continuous sub-sampled Hessian: For any subset, $S_n \subset [\mathcal{S}_n]$ *in* $n \in [N]$, *there exists constant* L_n, *such that* $\forall \boldsymbol{\theta}, \boldsymbol{\theta'} \in \mathbb{R}^d$,

$$||\boldsymbol{H}_{S_n}(\boldsymbol{\theta}) - \boldsymbol{H}_{S_n}(\boldsymbol{\theta'})||_2 \leq L_n ||\boldsymbol{\theta} - \boldsymbol{\theta'}||$$

where, $\boldsymbol{H}_{S_n}(\boldsymbol{\theta}) = \nabla^2 f_n(\boldsymbol{\theta})$ over the subset S_n.

The approach in [16] is based on a low-rank approximation of sub-sampled Hessians (Hessians obtained over the subset of data points of size S_n). The low-rank approximation of local sub-sampled hessian $\boldsymbol{H}_{S_n}(\boldsymbol{\theta})$ is performed by using a truncated singular value decomposition (SVD) along the direction of large eigenvalues only, as, $\hat{H}_{S_n^k} = \boldsymbol{U}_r^k(\Lambda_r^k)\boldsymbol{U}_r^{k^T}$, where Λ_r is $r \times r$ diagonal matrix with the r largest eigenvalues of the local sub-sampled hessian $\boldsymbol{H}_{S_n^k}$ and $\boldsymbol{U}_r$ is a $d \times r$ matrix whose columns correspond to the eigenvectors of $H_{S_n^k}$. The superscript k represents the iteration number.

For stability, the eigenvalues smaller than r^{th} largest eigenvalue are replaced by $(r+1)^{th}$ largest eigenvalue of $\hat{H}_n$, and the approximate inverse Hessian is then given by

$$\hat{H}^{-1}_{S_n^k} = \lambda_{r+1}^{k^{-1}} \boldsymbol{I}_d + \boldsymbol{U}_r^k(\Lambda_r^{k^{-1}} - \lambda_{r+1}^{k^{-1}} \boldsymbol{I}_r)\boldsymbol{U}_r^{k^T} \qquad (12)$$

The computational cost in classical second order methods is $\mathcal{O}(b_n d^2)$ operations per node for the local hessian and $\mathcal{O}(d^3)$ operations per node for the inverse of the local hessian. The NewSamp algorithm determines both the hessian and its inverse in $\mathcal{O}(b_n d + (|S_n|_{max} + r)d^2)$ operations per node [16].

The Algorithm: DANDA

Our proposed approximate Newton algorithm DANDA is based on the determinantal averaging algorithm, adapted to the analog transmission over a wireless fading MAC channel, as described in Algorithm 1. It takes a convergence parameter ϵ as input that is a threshold for convergence based on the fractional difference between two consecutive values of $\boldsymbol{F}(\boldsymbol{\theta}_k)$. The optimal PDF α^* [2] is the α value that takes minimum total power per node for convergence.

IV. Experimental Results

In this section, we provide numerical examples to illustrate the performance of the algorithms DNDA (with and without RLS), DANDA and their comparison with GBMA [5] and LocalNewton [12] (NOTE: In LocalNewton the local parameter estimates are updated L times using the Newton's algorithm before being communicated to the edge server using analog transmission for aggregation). We used the Million Song Dataset (MSD) [17] that has 90 audio features for 515,345 soundtracks to recognize the year of release of a song. We pre-process the data by omitting cases corresponding to the year below 1976 to avoid skewness and the outliers in the distribution. The features are normalized to have zero mean and unity variance and the response has zero mean [3]. We split the dataset into training and test sets based on the partition recommendation [18]. The simulated distributed network has an edge server and 100 nodes, with each node having a disjoint set of 4,200 data points obtained via uniform random sampling.

Parameter	Value
Network	
AWGN	$w_k \sim \mathcal{N}(\mu = 0, \sigma_w^2 = 4$ mW$)$
Channel Fading	$h_n \sim Rayleigh(\tilde{\sigma_h} = 1)$
Number of nodes	$N = 100$
DANDA	
PAF	7×10^4
Learning Rate	$\beta = 0.95$
Optimal PDF	$\alpha = 0.3$
Sub-Sample size	$\lvert S_n \rvert = 3000$
rank(sub-sampled hessian)	$r = 50$
DNDA	
PAF	8×10^{24}
Learning Rate	$\beta = 0.95$
Optimal PDF	$\alpha = 0.3$
LN	
PAF	2.5
Learning Rate	$\beta = 1$
L	1
GBMA	
PAF	0.3
Learning Rate	$\beta = 0.1$

TABLE II: Simulation Parameters

We considered a regularized least squares loss function (10) with regularization parameter, $\lambda = 0.1$. Table II shows the value of various simulation parameters for the experiments. The transmitted signals from the N nodes face an i.i.d Rayleigh fading channel gain with the distribution $p(|h|) = \frac{|h|}{\tilde{\sigma}_h^2} e^{\frac{-|h|}{2\sigma_h^2}}$, where $\mu_h = \tilde{\sigma_h}\sqrt{\frac{\pi}{2}}$ and $\sigma_h^2 = \frac{\tilde{\sigma_h^2}(4-\pi)}{2}$.

algorithm	average time per iteration per node
GBMA	0.009462 s
LN	0.04112 s
DNDA	0.04379 s
DANDA($r = 50, \lvert s_n \rvert = 3000$)	0.04084 s

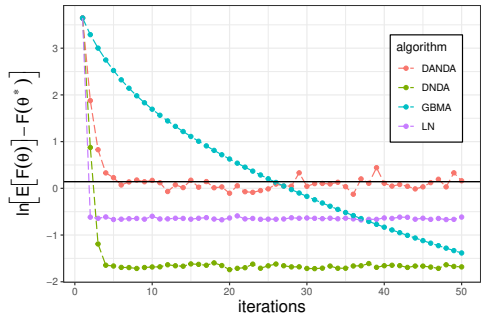
TABLE III: Average computer time (seconds) taken by algorithms per iteration per node on 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM

Fig. 2 shows a comparison of DANDA with LocalNewton (LN), GBMA and DNDA. The transmission parameters are set such that each algorithm takes approximately equal total energy for convergence[4]. In Fig. 2 (a) LN converges the fastest followed by DNDA and DANDA. The communication channel randomness and AWGN affects DANDA the most,

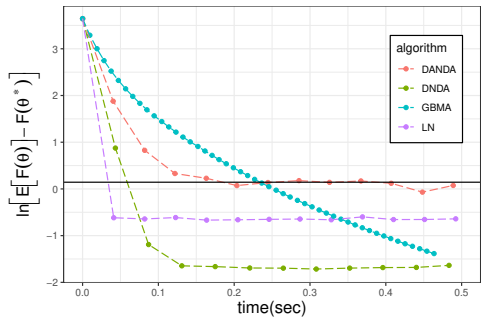
[2]We determined α^* by exhaustive search, varying α within a suitable range in $(0, 1)$, although more sophisticated methods can be used.

[3]This normalization method eliminates the need of intercept in linear regression model.

[4]The convergence criteria for the simulation is $\frac{|F(\boldsymbol{\theta}) - F(\boldsymbol{\theta^*})|}{F(\boldsymbol{\theta^*})} \leq 0.05$, where $F(\boldsymbol{\theta^*})$ is the optimal value of centralized newton method. The black horizontal line in all the Figures (except Fig. 5 & Fig. 6), shows this convergence threshold.

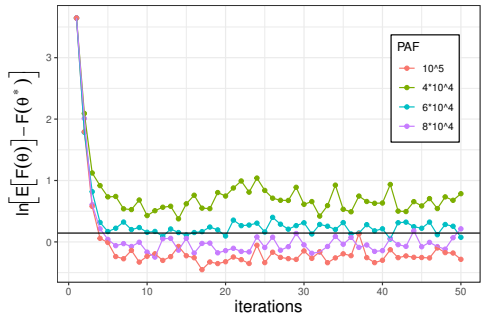


(a) Convergence performance of GBMA, LN, DNDA and DANDA

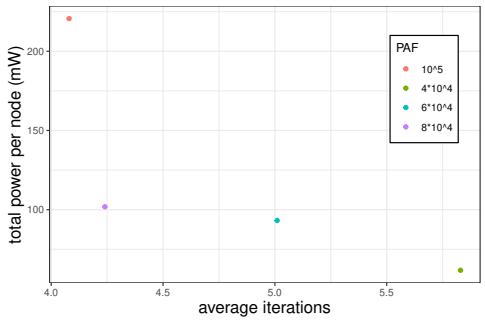


(b) Computation time comparison of GBMA, LN, DNDA and DANDA on 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM

Fig. 2: Simulation results on convergence and computation time averaged over 100 channel realizations



(a) Convergence performance of DANDA with different power adjustment factors



(b) Power requirements vs convergence rate performance of DANDA

Fig. 3: Simulation results for different PAF values for DANDA averaged over 100 channel realizations

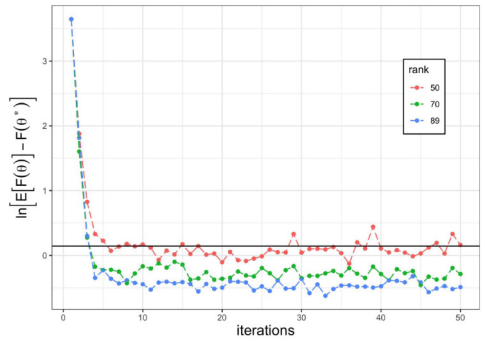


Fig. 4: Convergence comparison of DANDA with different rank values, keeping every other parameter constant

while DNDA and GBMA converge to the lowest value, GBMA requiring four times as many iterations to reach convergence. The Fig. 2 (b) shows the computation time taken to convergence by these algorithms. The GBMA takes the least time for computation per iteration as it is gradient based. The DANDA takes slightly shorter time for each iteration than LN and DNDA because it uses the approximate hessian and hence is computationally less expensive at nodes. Table III shows the average time per iteration (in seconds) required by the four algorithms. The computations are performed on 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM.

Fig. 3 shows the performance of the DANDA algorithm with different power settings, In Fig.3 (a), convergence improves with PAF[5], as increasing PAF effectively increases the signal-to-noise ratio (SNR). Fig.3 (b) shows the trade-off between total power required per node and average number of iterations the algorithm takes for convergence.

The DANDA algorithm uses hessian approximation by sub-sampling local data and using top r eigenvalues from the eigenvalue decomposition, indicated by the rank. Fig. 4 shows a comparison of DANDA with different rank values keeping $|S_n| = 3000$. As the rank values increases the hessian approximation moves closer to exact hessian and hence the performance of the algorithm improves.

Fig. 5 shows the accuracy on the test dataset comparison. All three algorithms based on second order optimization perform similarly on the test data, with LN showing the best results. The GBMA, once again, takes four times as many iterations to achieve its lowest mean squared error (MSE).

The total power consumed for convergence versus the average number of iterations required for DANDA in comparison with the DNDA, LN and GBMA is presented in Fig. 6. DANDA, DNDA and LN take similar number of iterations for a certain power consumption value [6], while GBMA clearly requires more iterations. Among the second order methods,

[5]The actual transmission power is the product of PAF and squared-norm of the vector to be transmitted, eg. in GBMA it is $P||\boldsymbol{g}_n(\boldsymbol{\theta_k})||_2^2$

[6]The power consumption values shown in Fig. 6 are approximate and comparable but not exactly identical for all algorithms due to the random channel gain and noise which affects $\boldsymbol{g}_n(\theta_k)$ and $\boldsymbol{H}_n(\theta_k)$ at every k^{th} iteration. The results shown in Fig. 6 are averaged over 100 channel realizations

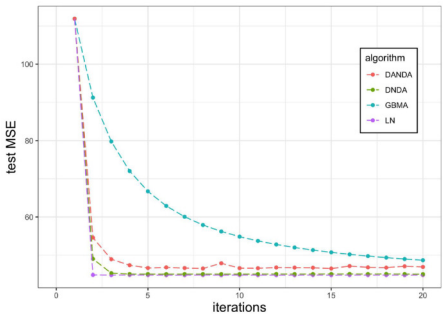


Fig. 5: Test data performance of DANDA, DNDA, GBMA and LN

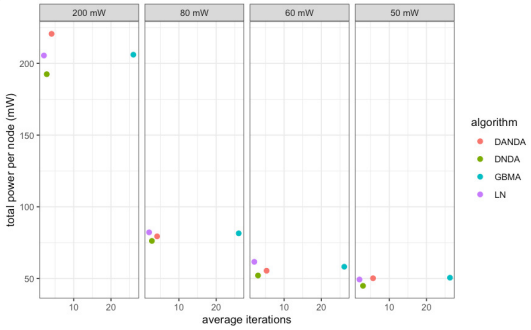


Fig. 6: Total power vs average iterations comparison of DANDA, DNDA, LN and GBMA

LN takes the minimum no. of iterations for a given power consumption value and DANDA takes the maximum, since LN transmits a single vector whereas DNDA (and DANDA) transmits two signals that makes them more prone to noise.

Fig. 7 shows a comparison of DNDA with DNDA based on RLS, and a combination of both. The recursive least squares estimation performs better compared to DNDA asymptotically but the convergence is slower. We used a combination of DNDA and RLS based DNDA where after the first iteration the algorithm switches from DNDA to the RLS based DNDA. This method is useful as the RLS based DNDA can be initialized by the parameter values obtained after the first iteration of DNDA, rather than with an arbitrary value.

V. Conclusions

We introduced a distributed ML algorithm, DANDA, that uses the Determinantal Averaging combined with a low-rank Hessian approximation at each node, and performs over-the-air edge learning. This algorithm provides the benefit of fast convergence and uses approximation methods to reduce the computation cost per iteration. The results show that the convergence performance of DANDA is much faster than state-of-the-art gradient based GBMA and compares favourably with true hessian based LN in terms of total power consumption and computational time, with better privacy guarantee as it does not transmits the parameter vector, $\boldsymbol{\theta}$ itself.

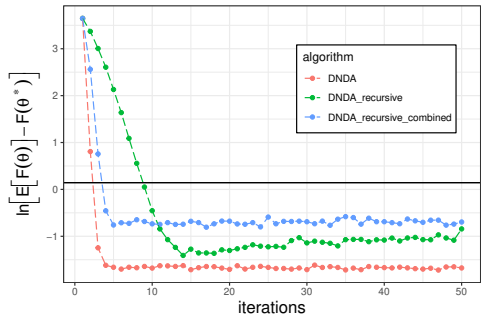


Fig. 7: Convergence performance of DNDA, DNDA with RLS and a combination of both

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data", *Artificial Intelligence and Statistics PMLR*, 2017, pp. 1273–1282.

[2] Mohammad Mohammadi Amiri and Deniz Gündüz, "Over-the-air machine learning at the wireless edge", *Proc.IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2019.

[3] Guangxu Zhu, Yong Wang and Kaibin Huang, "Broadband analog aggregation for low-latency federated edge learning", *IEEE Transactions on Wireless Communications*, vol. 19, pp. 491–506, January 2019.

[4] Mohammad Mohammadi Amiri and Deniz Gündüzuz, "Federated learning over wireless fading channels". *IEEE Transactions on Wireless Communications* vol. 19, pp. 3546–3557, May 2020.

[5] Tomer Sery and Kobi Cohen, "On analog gradient descent learning over multiple access fading channels", *IEEE Transactions on Signal Processing*, vol. 68, pp. 2897–2911, 2020.

[6] Huayan Guo, An Liu, and Vincent KN Lau, "Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis", *IEEE Internet of Things Journal*, vol. 8, pp. 197–210, Jan 2020.

[7] Tomer Sery, Nir Shlezinger, Kobi Cohen and Yonina C. Eldar, "Over-the-air federated learning from heterogeneous data", *IEEE Transactions on Signal Processing*, vol. 69, pp. 3796 - 3811, June 2021.

[8] Ohad Shamir, Nati Srebro and Tong Zhang, "Communication-efficient distributed optimization using an approximate Newton-type method". *International Conference on Machine Learning (PMLR)*, pp. 1000–1008, 2014.

[9] Canh T Dinh, Nguyen H. Tran, Tuan Dung Nguyen and Wei Bao, "DONE: Distributed Newton-type Method for Federated Edge Learning", *IEEE Transactions on Parallel and Distributed Systems*, early access, DOI: 10.1109/TPDS.2022.3146253.

[10] Shusen Wang, Farbod Roosta-Khorasani and Michael W. Mahoney, "Giant: Globally improved approximate newton method for distributed optimization", *Advances in Neural Information Processing Systems 31*, pp. 2332–2342, 2018.

[11] Yuchen Zhang and Xiao Lin, "DiSCO: Distributed optimization for self concordant empirical loss",*Proc. International Conference on Machine Learning (PMLR)*, pp. 362–370, 2015.

[12] Vipul Gupta et al, "LocalNewton: Reducing communication rounds for distributed learning". *Proc. Uncertainty in Artificial Intelligence (PMLR)*, pp. 632–642, 2021.

[13] Sheng Hua, Kai Yang and Yuanming Shi, "On-device federated learning via second-order optimization with over-the-air computation", *Proc. 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5, 2019.

[14] M. Krouka, A. Elgabli, C. B. Issaid and M. Bennis, "Communication-Efficient Federated Learning: a Second Order Newton-type Method with Analog Over-the-Air Aggregation," in IEEE Transactions on Green Communications and Networking, doi: 10.1109/TGCN.2022.3173420.

[15] Michal Derezinski and Michael W Mahoney, "Distributed estimation of the inverse Hessian by determinantal averaging", *Advances in Neural Information Processing Systems 32*, 2019.

[16] Murat A Erdogdu and Andrea Montanari, "Convergence rates of sub sampled Newton methods", *Advances in Neural Information Processing Systems 28*, 2015.

[17] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman and Paul Lamere, "The Million Song Dataset", *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[18] Dheeru Dua and Casey Graff, *UCI Machine Learning Repository*, 2017, URL: http://archive.ics.uci.edu/ml.