Decentralized Stochastic Optimization with Gossip and Compressed Communication

-

April 2023

1 Introduction

Decentralised machine learning is increasingly becoming the core of many crucial applications, both in terms of scalability to larger datasets and systems and in terms of the localization, ownership, and privacy of the data. Decentralised optimisations include multiple participating devices as opposed to the case of a single central controller with large computational capability. Such optimisation necessitates computations of gradients or sub-gradients on the device and local communications between neighbouring devices. These training sessions are now conducted directly on consumer electronics, which always maintain the confidentiality of their portion of the data.

Generally. a distributed optimization problem spread across n devices/nodes has the goal of finding,

$$f^* = \min x \in \mathbb{R}^d \left[f(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \tag{1}$$

Decentralized Communication The network topology can be modelled by a graph where the edges represent the communication links along which the messages are transmitted and received between a pair of nodes (eg, model updates). In a centralized network, there are significant bottlenecks on the central node in terms of communication latency, bandwidth and fault tolerance. Decentralized topologies avoid these bottlenecks and thereby offer hugely improved potential in scalability. The main advantage of decentralized algorithms over centralized algorithms lies on avoiding the communication traffic in the central node. In particular, decentralized algorithms could be much more efficient than centralized algorithms when the network bandwidth is small and the latency is large. For example, while the master node in the centralized setting receives (and sends) in each round messages from all workers, $\Theta(n)$ in total, in decentralized topologies the maximal degree of the network is often constant (e.g. ring or torus) or a slowly growing function in n (e.g. scale-free networks).

We consider a network in terms of a $G = (V, E)$ of $V = \{1, ..., n$ set of nodes and E edges, where each $(i, j) \in E$ is an unordered pair of distinct nodes. The set of neighbours of node i is denoted by $N_i = \{j \in V : (i, j) \in E\}$. A pair of agents/devices can exchange data if and only if they are connected by a direct communication link. The graph is assumed to be undirected and we can assume any node to be the head or the tail randomly (with equal probability).

Average Consensus In networks of agents (or dynamic systems), "consensus" means to come to an understanding regarding a specific amount of interest that is dependent upon the state of all agents. A "consensus algorithm" (or protocol) is an algorithm that specifies the information exchange between an agent and all of its neighbors on the network so that they reach consensus in a finite amount of time. Let x_i denote the value of node v_i. This value of the node can represent any physical quantity (as in a sensor network) like, temperature, humidity and so on. Two nodes v_i and v_j are said to be in agreement if and only if $x_i = x_j$. The whole network is said to be in consensus when $x_i = x_j$ for all $i, j \in V, i \neq j$. This final value agreed upon is called the group decision value α. A consensus algorithm finds an agreement state between these n agents. There are several cases for the consensus problems based on this group decision value. The average consensus is the case of the consensus problem where the agreement value is the average of the initial states of all the agents/devices, i.e., $x^* = \frac{1}{n} \sum_{i=1}^{n} x_i(0)$

2 Gossips Algorithm for Average Consensus

The problem of average consensus is solved using a a type of decentralized algorithm known as the gossips algorithm. A general form of this classic algorithm that generates sequence $x_i^{(t)}$ on every node iteratively is,

$$x_i^{(t+1)} = x_i^{(t)} + \gamma \sum_{j=1}^{n} w_{ij} \Delta_{ij}^{(t)} \tag{2}$$

Here γ is the step size parameter between 0 and 1, $w_{ij} \in [0, 1]$ are the averaging weights and $\Delta_{ij}^{(t)}$ is the information vector that is sent from node i to node j at the time instant t. The gossip or interaction matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ is defined as $(\mathbf{W})_{ij} = w_{ij}$ and it affects the convergence rate of the gossip algorithm.

The exact gossip algorithm corresponds to the below choice in the general equation (2),

$$\gamma = 1 \qquad \Delta_{ij} = x_j^{(t)} - x_i^{(t)} \tag{3}$$

i.e.,

$$x_i^{(t+1)} = x_i^{(t)} + \sum_{j=1}^{n} w_{ij}(x_j^{(t)} - x_i^{(t)}) \tag{4}$$

The dynamics of this exact gossip system as a whole can be written as,

$$\mathbf{x}^{(t+1)} = \mathbf{W}\mathbf{x}^{(t)}. \tag{5}$$

2.1 Convergence of average consensus

It is clear from eq (5) that $\mathbf{x}^{(t)} = \mathbf{W}^t\mathbf{x}^{(0)}$. The $\mathbf{W}$ matrix must be chosen such that, for any initial value $\mathbf{x}^{(0)}$, $\mathbf{x}^{(t)}$ converges to the average vector. i.e.,

$$\overline{\mathbf{x}} = (\mathbf{1}^{\mathrm{T}}\mathbf{x}^{(0)}/n)\mathbf{1} \tag{6}$$

i.e.,

$$\lim_{t\to\infty} \mathbf{x}^{(t)} = \lim_{t\to\infty} \mathbf{W}^t\mathbf{x}^{(0)} \tag{7}$$

$$= \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\mathbf{x}^{(0)} \tag{8}$$

This is equivalent to the matrix equation,

$$\lim_{t\to\infty} \mathbf{W}^t = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \tag{9}$$

2.1.1 Conditions for Convergence

From above, the gossip algorithm for average consensus converges for any initial vector if and only if the below matrix equation holds,

$$\lim_{t\to\infty} \mathbf{W}^t = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \tag{10}$$

For the above equation to hold, the necessary and sufficient conditionn are presented below.

$$\mathbf{1}^{\mathrm{T}}\mathbf{W} = \mathbf{1}^{\mathrm{T}} \tag{11}$$

$$\mathbf{W}\mathbf{1} = 1 \tag{12}$$

$$\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) < 1 \tag{13}$$

where $\rho()$ denotes the spectral radius of a matrix. (Maximum of the absolute value of the eigenvalues of the matrix.)

The interpretation of the above conditions is as follows,

- Eq. (11) and eq. (12) mean that $\mathbf{W}$ is row and column stochastic, respectively. If we assume the elements of $\mathbf{W}$ are non-negative, eq. (11) and eq. (12) together state that $\mathbf{W}$ is a double stochastic matrix.
- Eq. (13) means that 1 is a simple eigenvalue of $\mathbf{W}$ and all the other eigenvalues are less than one (magnitude wise).
- It should also be noted that $\rho(A) \leq ||A||_2$ for any matrix $\mathbf{A}$, which means the spectral radius of a matrix is always less than equal to the spectral norm. The equality holds when matrix $\mathbf{A}$ is symmetric.

2.2 The Weight/Mixing Matrix

In this part, we see how to calculate the Weight/Mixing (**W**) Matrix. First we define the Laplacian of the graph and other useful matrices.

2.2.1 Laplacian of a graph

The degree matrix of the graph is an $n \times n$ matrix **D** defined as **D**(G) where $\mathbf{D}_{ij}$ equals,

$$\mathbf{D}_{ij} = \begin{cases} deg(v_i), & \text{if } i = j \\ 0, & otherwise \end{cases}$$

Let **A** denote the adjacency matrix of the graph where,

$$\mathbf{A}_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & otherwise \end{cases}$$

Then the Laplacian of the graph is defined as,

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{14}$$

An important feature of **L** is that all the row sums of **L** are zero and hence $e_0 = (1, 1, ..., 1)^T \in \mathbb{R}^n$ is an eigenvector of **L** associated with eigenvalue $\lambda = 0$. There is an alternate way of defining the Laplacian matrix. First we define an *incidence matrix* which is an $n \times n$ matrix as $\mathbf{C} = [c_{ij}]$ where,

$$c_{ij} = \begin{cases} +1, & \text{if } v_i \text{ is the head of the edge } e_j \\ -1, & \text{if } v_i \text{ is the tail of the edge } e_j \\ 0, & otherwise \end{cases}$$

The Laplacian matrix follows the property $\mathbf{L} = \mathbf{C}\mathbf{C}^{\mathrm{T}}$. This property holds regardless of the orientation of the graph. We will see the use of this property later on.

2.2.2 Choice of W matrix

The weights of the Gossip/Interaction (**W**) matrix is chosen according to,

$$\mathbf{W} = \mathbf{I} - \epsilon\mathbf{L} \tag{15}$$

where **I** is an $n \times n$ identity matrix and ϵ is a constant. More clearly the elements of this matrix are,

$$\mathbf{W}_{ij} = \begin{cases} \epsilon, & \text{if } \{i,j\} \in E \\ 1 - D_{ij}\epsilon, & \text{if } i = j \\ 0, & otherwise \end{cases}$$

2.2.3 Choice of ϵ

Since **L** is positive semi-definite, we must have $\epsilon > 0$ for the convergence condition $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) < 1$ to hold. From eq. (15), the eigenvalues of **W** in terms of those of **L** are,

$$\lambda_i(\mathbf{W}) = 1 - \epsilon\lambda_{n-i+1}(\mathbf{L}) \tag{16}$$

for $i = 1, .., n$. where $\lambda_i()$ denotes the ith largest eigenvalue of a symmetric matrix. The spectral radius of $(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n)$ is then expressed as,

$$\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) = \max(\lambda_2(\mathbf{W}), -\lambda_n(\mathbf{W})) \tag{17}$$

$$= \max(1 - \epsilon\lambda_{n-1}(\mathbf{L}), \epsilon\lambda_1(\mathbf{L}) - 1) \tag{18}$$

Proof why $\lambda_2(\mathbf{W})$ is the largest eigenvalue of W

According to the Weilandt deflation method [1] if the eigenvalues of a matrix **A** are known lets say, $\{e_1, ...e_n\}$, then the eigenvalues of a matrix of form below (known as *deflated matrix* of the original matrix) can be calculated,

$$\mathbf{A}_1 = \mathbf{A} - \sigma u_1 v^H \tag{19}$$

where u_i is the eigenvector corresponding the largest eigenvalue of matrix $\mathbf{A}$, v is any arbitrary vector such that $v^H u_1 = 1$. Then, the eigenvalues of $\mathbf{A}_1$ are the same as those of $\mathbf{A}$ except for the eigenvalue e_1 which now becomes $e_1 - \sigma$.

The matrix $\mathbf{W} - \mathbf{1}\mathbf{1}^T/n$, is the *deflated matrix* of $\mathbf{W}$ with $u_1 = \mathbf{1}$, $v^T = \frac{1}{n}\mathbf{1}^T$ and $\sigma = 1$, Then according to the Weilandt deflation method, the corresponding eigen value of the deflated matrix is $\lambda_1(\mathbf{W}) - 1$ while all the other eigen values are same as that of $\mathbf{W}$. As we know $\lambda_1(\mathbf{W}) = 1$, therefore the corresponding eigen value of deflated matrix is 0. This means that the largest eigenvalue of $\mathbf{W} - \mathbf{1}\mathbf{1}^T/n$ is either $\lambda_2(\mathbf{W})$ or $-\lambda_n(\mathbf{W})$,whichever is higher in magnitude.

From the 17 we can define the range of ϵ over which convergence is guaranteed. $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^T/n) < 1$, if and only if,

$$0 < \epsilon < \frac{2}{\lambda_1(\mathbf{L})} \tag{20}$$

The choice of ϵ that minimizes eq.(17) is,

$$\epsilon^* = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{n-1}(\mathbf{L})} \tag{21}$$

In particular, convergence is guaranteed as long as $\epsilon \in (0, \frac{1}{d_{max}})$, where d_{max} is the maximum degree over all the nodes in the graph topology.

3 Gossip with Compressed Communication

In each iteration of the exact gossip algorithm, a full dimensional vector is exchanged between two neighbouring nodes for every link on the communication topology, i.e., a node j sends $x_j^{(t)}$ to all its neighbours i for all $j \in E$. This is computationally expensive and makes the communication slowers. To reduce the overall complexity of this communication, this vector g is compressed before sending, denoted as $Q(g)$ for a compression operator $Q : \mathbb{R}^d \longrightarrow \mathbb{R}^d$. There are two ways to compress vector. First is by introducing sparsity and the second way by quantization, i.e., by reducing the number of bits required to represent g. The paper assumes that the compression operator $Q : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ satisfies,

$$\mathbb{E}_Q ||Q(\mathbf{x}) - \mathbf{x}||^2 \leq (1 - \delta)||\mathbf{x}||^2 \tag{22}$$

for all $\mathbf{x} \in \mathbb{R}^d$ and parameter $\delta > 0$. $\mathbb{E}_{\mathbb{Q}}$ is the expectation over the internal randomness of operator Q. Example of this compression operation

- sparsification (also known as gradient masking): Randomly selecting k out of d coordinates ($rand_k$) or the k coordinates with the highest magnitude values (top_k) [2].
- randomized gossip: Setting $\mathbf{Q(x)} = \mathbf{x}$ with a probability $p \in (0, 1]$ and $\mathbf{Q(x)} = 0$ otherwise.
- rescaled unbiased estimator: suppose $\mathbb{E}_{\mathbb{Q}} Q(x) = \mathbf{x}$ for all $x \in \mathbb{R}^d$ and $\mathbb{E}_Q ||Q\mathbf{x}||^2 \leq \tau ||\mathbf{x}||^2$, then $Q^{'}(\mathbf{x}) = \frac{1}{\tau} Q(\mathbf{x})$ satisfies eq. (22) with $\delta = \frac{1}{\tau}$.
- random quantization: For precision levels $s \in \mathbb{N}_+$ and $\tau = \left(1 + min\left\{\frac{d}{s^2}, \frac{\sqrt{d}}{s}\right\}\right)$ the quantization operator

$$qsgd_s(x) = \frac{sign(x).||\mathbf{x}||}{s\tau} \times \left[s.\frac{|\mathbf{x}|}{||\mathbf{x}||} + \zeta\right] \tag{23}$$

  for random variable $\zeta \sim_{u.a.r} [0, 1]^d$ satisfies with eq. (22) $\delta = \frac{1}{\tau}$

3.1 Communication-Efficient Methods for Average Consensus

Scheme 1 : Aysal et al. (2008) propose the quantized gossip (Q1-G),

$$\gamma = 1, \qquad \Delta_{ij}^{(t)} = Q(\mathbf{x}_j^{(t)}) - \mathbf{x}_i^{(t)}, \tag{Q1-G}$$

in the general equation (2), i.e., to apply the compression operator directly on the message that is send out from node j to node i. However, this algorithm does not preserve the average of the iterates over the iterations,

$\frac{1}{n}\sum_{i=1}^{n} x_i^{(0)} \neq \frac{1}{n}\sum_{i=1}^{n} x_i^{(t)}$ for the $t \geq 1$, and as a consequence does not converge to the optimal solution, $\overline{x}$.

Scheme 2: An alternative proposal by Carli et al. (2007) alleviates this drawback. The scheme

$$\gamma = 1, \qquad \Delta_{ij}^{(t)} = Q(\mathbf{x}_j^{(t)}) - Q(\mathbf{x}_i^{(t)}), \tag{Q2-G}$$

preserves the average of the iterates over the iterations. However, the scheme also fails to converge for arbitrary precision. If $\overline{x} \neq 0$ the noise introduced by the compression, $||Q(x_j^{(t)})||$ does not vanish for $t \to \infty$. As a consequence, the iterates oscillate around $\overline{\mathbf{x}}$ when the compression error becomes larger than the suboptimality $||x_i^{(t)} - \overline{x}||$

Scheme 3: The choice for Δ and the communication step for the Choco Gossip Algorithm, proposed in [3], is summarized as,

$$\Delta_{ij}^{(t)} = \hat{\mathbf{x}}_j^{(t)} - \hat{\mathbf{x}}_i^{(t)} \tag{24}$$

$$\hat{\mathbf{x}}_j^{(t+1)} = \hat{\mathbf{x}}_j^{(t)} + Q(\mathbf{x}_j^{(t+1)} - \hat{\mathbf{x}}_j^{(t)}) \tag{25}$$

Here $\hat{\mathbf{x}}_i^{(t)} \in \mathbb{R}^d$ denotes additional variables that are stored by all neighbours j of node i, $\{i, j\} \in E$ as well as on node i itself. The value of γ is chosen according to the type of compression technique used.

The decentralized Choco Gossip algorithm is shown below.

Algorithm 1: CHOCO-GOSSIP

Input: Initial values $x_i^{(0)} \in \mathbb{R}$ on each node $i \in [n]$, stepsize γ, network graph $\mathrm{G} = ([n], E)$ and mixing matrix $\mathbf{W}$, initialize $\hat{\mathbf{x}}_i^{(0)} := 0 \forall i$
for t ***in*** $[0, T-1]$ **do**
  $\hat{a}_{ij} := \hat{\mathrm{x}}_j^{(t)} - \hat{\mathrm{x}}_i^{(t)}$
  $\mathrm{x}_i^{(t+1)} := \mathrm{x}_i^{(t)} + \gamma \sum_{j:\{i,j\}\in E} w_{ij}\hat{a}_{ij}$
  $\mathrm{q}_i^{(t)} := Q(\mathrm{x}_i^{(t+1)} - \hat{x}_i^{(t)})$
  for *neighbours j :* $\{i, j\} \in E$ *(including* $\{i\} \in E$*)* **do**
    send $\mathrm{q}_i^{(t)}$ and receives $\mathrm{q}_j^{(t)}$
    $\hat{\mathrm{x}}_j^{(t+1)} := \hat{\mathrm{x}}_j^{(t)} + \mathrm{q}_j^{(t)}$
return

4 Consensus-based Decentralized Stochastic Gradient Descent (D-SGD) Algorithm

4.1 Decentralized Stochastic Optimization

The idea behind Choco average consensus is extended to achieve consensus among the nodes in a decentralized optimization setting with communication restrictions. The decentralized optimization presented in (1) allows for different local objectives f_i. These local objectives are set to have stochastic optimization structure such that each nodes solves the gradient descent problem.

$$f_i(\mathbf{x}) := \mathbb{E}_{\zeta_i \sim \mathbb{D}_i} F_i(\mathbf{x}, \zeta_i) \tag{26}$$

where F_i is the individual loss functions at node i and $\mathbb{D}_i$ are the data distributions on each node. This data distribution need not be same for all the nodes and can be different at each node.

The optimization problem that the algorithm solves is,

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{n}\sum_{i}^{n} \underbrace{\mathbb{E}_{\zeta \sim \mathbb{D}_i} F_i(\mathbf{x}; \zeta)}_{f_i(\mathbf{x})} \tag{27}$$

For analyzing the decentralized algorithm, we first make some common optimization assumptions [4], [5].
Assumptions on the local objective functions

- *All the functions $F_i()$'s are are L-smooth and μ strongly convex.*
- *The variance on each worker*

$$\mathbb{E}_{i\sim U([n])}\mathbb{E}_{\zeta\sim\mathbb{D}_i}||\nabla F_i(\mathbf{x};\zeta)-\nabla f(\mathbf{x})||^2$$

is bounded for any x with i uniformly sampled from [1, 2, . . ., n] and ζ from the distribution $\mathbb{D}_i$. This implies that there exists constants a, b such that,

$$\mathbb{E}_{\zeta\sim\mathbb{D}_i}||\nabla F_i(\mathbf{x};\zeta)-\nabla f(\mathbf{x})||^2\leq a^2,\forall i,\forall\mathbf{x}$$
$$\mathbb{E}_{i\sim U([n])}||\nabla f_i(\mathbf{x})-\nabla f(\mathbf{x})||^2\leq b^2,\forall\mathbf{x}$$

Here ∇ is the gradient operator, applied to the local loss function (F_i), local objective function (f_i) and the global objective function (f)

4.2 Communication-Efficient Method for Decentralized SGD

In this part we talk about the already existing methods that are there to solve DSGD.

4.2.1 Difference Compression D-SGD (DCD-SGD)

In this method presented in [4], the nodes exchange the compressed difference of local models between two successive iterations, instead of exchanging the local models. The algorithm is defined as,

- take the weighted average and apply stochastic gradient descent step, i.e., $x_i^{t+\frac{1}{2}}=\sum_{i=1}^n w_{ij}\hat{x}_h^{(t)}-\eta\nabla F_i(x_i(t);\xi_i^{(t)})$. Here $\hat{x}_j^{(t)}$ is the replica of $x_j^{(t)}$ but is stored on node i_3
- compress the difference between $x_i^{(t)}$ and $x_j^{(t+\frac{1}{2})}$ and update the local model $z_i^{(t)}=x_i^{(t+\frac{1}{2})}-x_i^{(t)}$, $x_i^{(t+1)}=x_j^{(t)}+Q(z_i^{(t)})$
- Send this updated model to the neighbours and recieve their updated models.

4.2.2 Extrapolation Compression D-SGD(ECD-SGD)

In this method (also proposed in [4]), instead of sending the local model directly $x_i^{(t)}$ directly to the neighbouring nodes, a *z*- value is sent. This *z*-value is extrapolated from $x_i^{(t)}$ and $x_i^{(t-1)}$ at each iteration. Each node (let's say *i*) estimates its neigbour;s values $x_j^{(t)}$ and compressed *z*-value at the *t*-th iteration. More clearly at the *t*th iteration, node *i* performs the following steps to estimate $x_k^{(t)}$ by the use of $\overline{x}_j^{(t)}$

- the node *j, computes the t-value that is obtained through the extrapolation*

$$z_j^{(t)}=(1-0.5t)x_j^{(t-1)}+0.5tx_j^{(t)} \tag{28}$$

- *Compress $z_j^{(t)}$ and send it to its neighbours, say node i which computes $\overline{x}_j^{(t)}$ using*

$$\overline{x}_j^{(t)}=(1-2t)\overline{x}_j^{(t-1)}+2t^{-1}Q(z_j^{(t)}) \tag{29}$$

4.2.3 CHOCO-SGD

Below is the algorithm for CHOCO-SGD which is proposed in [3]. This algorithm consists of four parts. Each node calculates the stochastic gradient descent (line 3), then the values of the nodes are updated based on the weighted average of the neighbours (line 4), then the vector to be sent is compressed using the compression operation in line 5, and at the end this vector is sent to all the neighbouring nodes (line 6-9)

Algorithm 2: CHOCO-SGD

Input: Initial values $x_i^{(0)} \in \mathbb{R}$ on each node $i \in [n]$, consensus stepsize γ, SGD stepsize $\{\eta_t\}_{t\geq 0}$, network graph $\mathrm{G} = ([n], E)$ and mixing matrix $\mathbf{W}$, initialize $\hat{\mathbf{x}}_i^{(0)} := 0 \; \forall i$
for t ***in*** $[0, T-1]$ **do**
  Sample $\xi_i^{(t)}$, compute gradient $\mathrm{g}_i^{(t)} := \nabla F_i(x_{(i)}^t, \xi_i^{(t)})$
  $\mathrm{x}_i^{(t+\frac{1}{2})} := \mathrm{x}_i^{(t)} - \eta_t \mathrm{g}_i^{(t)}$
  $\hat{a}_{ij} := \hat{\mathrm{x}}_j^{(t)} - \hat{\mathrm{x}}_i^{(t)}$
  $\mathrm{x}_i^{(t+1)} := \mathrm{x}_i^{(t+\frac{1}{2})} + \gamma \sum_{j:\{i,j\}\in E} w_{ij}\hat{a}_{ij}$
  $\mathrm{q}_i^{(t)} := Q(\mathrm{x}_i^{(t+1)} - \hat{x}_i^{(t)})$
  for *neighbours j :* $\{i, j\} \in E$ *(including* $\{i\} \in E$*)* **do**
    send $\mathrm{q}_i^{(t)}$ and receives $\mathrm{q}_j^{(t)}$
    $\hat{\mathrm{x}}_j^{(t+1)} := \hat{\mathrm{x}}_j^{(t)} + \mathrm{q}_j^{(t)}$
return

5 New Ideas

5.1 Random Activation with Constant Bits

To decrease the communication cost instead of transmitting data from all the nodes, we activate only a certain number of nodes (less than the total nodes) in the network. We choose *k* nodes out of the *n* nodes to broadcast data to their neighbours using $floor(\frac{bn}{k})$ bits where *b* is the minimum quantization bit allowed for a single node. The total bits alloted to the network are *bn*. The total bits used by this system are equal to or less than the total bits alloted to the network.

5.1.1 System Model

Let us have a family of independent binary random variable $\mathbf{A}_{ij}, i, j = 1, .., N, i \neq j$ such that,

$$\mathbb{P}[\mathbf{A}_{ij} = 1] = p, \qquad \mathbb{P}[\mathbf{A}_{ij} = 0] = 1 - p$$

. Since we assume that a node always has knowledge of it's own data, therefore,

$$\mathbf{A}_{ii} = 1$$

The above means that $\mathbf{A}_{ij}$ is one with a probability *p* and 0 with a probability of (1-*p*). This means the edge between node *i* and node *j* is activated with a probability of *p*. The sum $(\sum_{i=1}^{N}\sum_{j=i+1}^{N} M_{ij} = K) < N$ which signifies that in every round, which signifies that in every round, K links are activated on average for the data to be transmitted (sent or received) using it.

Since in every iteration not all nodes are selected from transmission, we have to make modifications to the Mixing (**W**) matrix. The bias compensation method proposed in [6] is used to compensate for the missing information. The new mixing matrix is obtained using the following modification,

$$\overline{w}_{ij} = \begin{cases} w_{ij}a_{ij}, & \text{if } j \neq i \\ 1 - \sum_{k=1}^{n} w_{ik}a_{ik}, & \text{if } j = i \end{cases}$$

Then the average consensus algorithm takes the following form,

$$x_i^{(t+1)} = x_i^{(t)} + \sum_{j=1}^{n} \overline{w}_{ij}(x_j^{(t)} - x_i^{(t)}) \tag{30}$$

If we define matrices **D** and **Q** as,

$$\mathbf{Q}_{ij} = \begin{cases} 0, & \text{if } j = i \\ a_{ij}w_{ij}, & \text{if } j \neq i \end{cases}$$

$$\mathbf{D}_{ij} = \begin{cases} w_{ii} + 1 - \sum_{j \neq i} w_{ih} a_{ih}, & \text{if } j = i \\ 0, & \text{if } j \neq i \end{cases}$$

Then the above consensus can be written in matrix form as,

$$\mathbf{x}^{(t+1)} = \overline{\mathbf{W}} \mathbf{x}^{(t)} \tag{31}$$

where

$$\overline{\mathbf{W}} = \mathbf{D} + \mathbf{Q} \tag{32}$$

This bias compensation method ensures convergence of the algorithm. In general the consensus value is not the average of the initial values but a scaled average of the initial values. This is due to the fact that the new mixing matrix $\overline{\mathbf{W}}$ is only row stochastic and not necessarily doubly stochastic.

6 Dataset

The experiments are done using the epsilon dataset which is a binary classification dataset [7]. The performance is assesed on the logistic regression, defined as $\frac{1}{m} \sum_{j=1}^{m} log(1 + \exp(-b_j \mathbf{a}_j^T \mathbf{x})) + \frac{1}{2m} ||\mathbf{x}||^2$, where $(a_j) \in \mathbb{R}^d$ and $b_j \in \{-1, 1\}$ are the data samples and m denotes the number of samples in the dataset.

7 Experiments

The experiments have been divided into four parts and each of these are studied on the below graph type,

- Ring Topology
- Star topology
- Erdos-Renyi topology (Random)

7.1 Part 1: Performance of Plain Decentralized SGD

In this part, we test the performance of the plain decentralized SGD for different number of nodes. The subotimality reduces as the number of nodes reduces. A slightly faster convergence is noted as the number of nodes is increased.

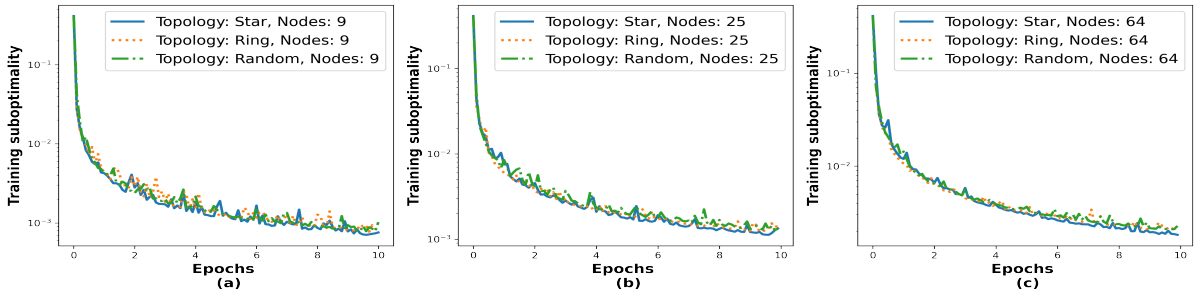

Figure 1: Comparison of Plain SGD algorithm for different Topologies.

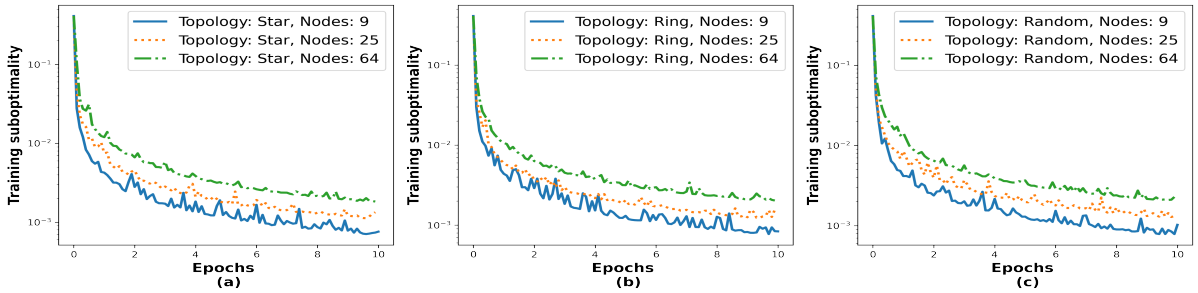

Figure 2: Comparison of Plain SGD algorithm for different number of nodes.

7.2 Part 2: Performance of DCD-PSGD

In this part, we have introduced unbiased compression with random quantization to the DCD-PSGD algorithm. We have then compared the performance for different number of quantization bits. The number of quantization bit to be studied, are [2, 4, 8]. We see that 2 bit quantization is not useful at all in exchanging the information. The algorithm is not able to converge and moreover the loss quickly becomes infinity. Fig. 3 shows the comparison of DCD-PSGD algorithm for different number of nodes and we see a quicker convergence for larger number of nodes.

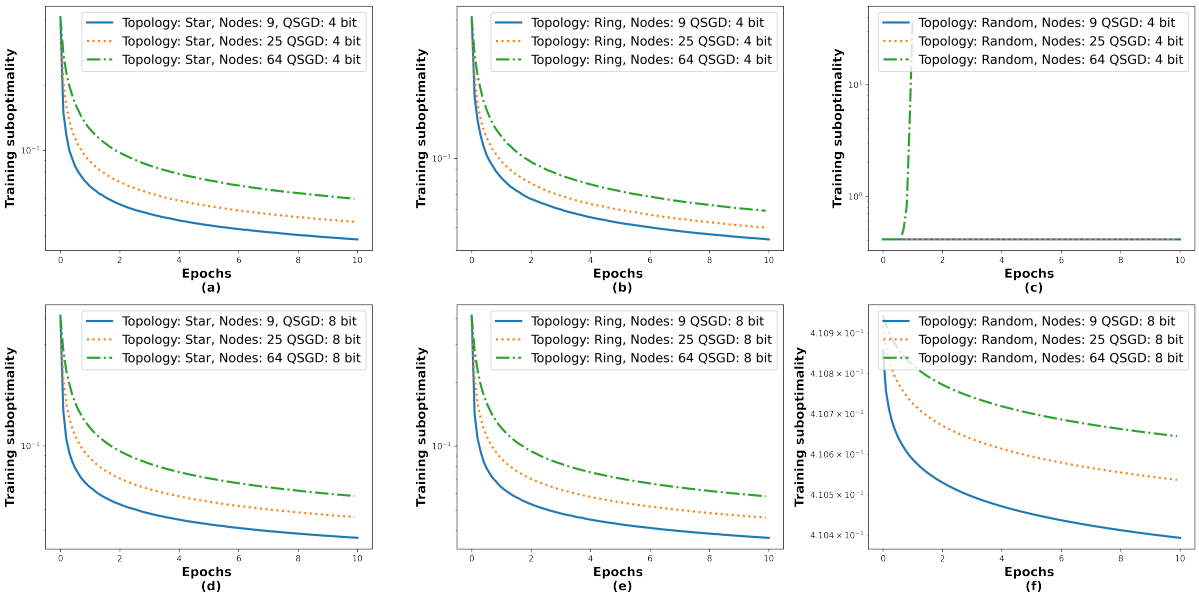

Figure 3: Comparison of DCD-PSGD for different number of nodes.

Fig. 4 shows the comparison of DCD-PSGD algorithm for different number of quantization bits. it is clear that 2 bit quantization is not useful at all. Quantization with 8 bits performs slightly better or equal in most of the cases compared to with 4 bits. For better performance we should choose 8 bit quantization but it comes at a computational cost.

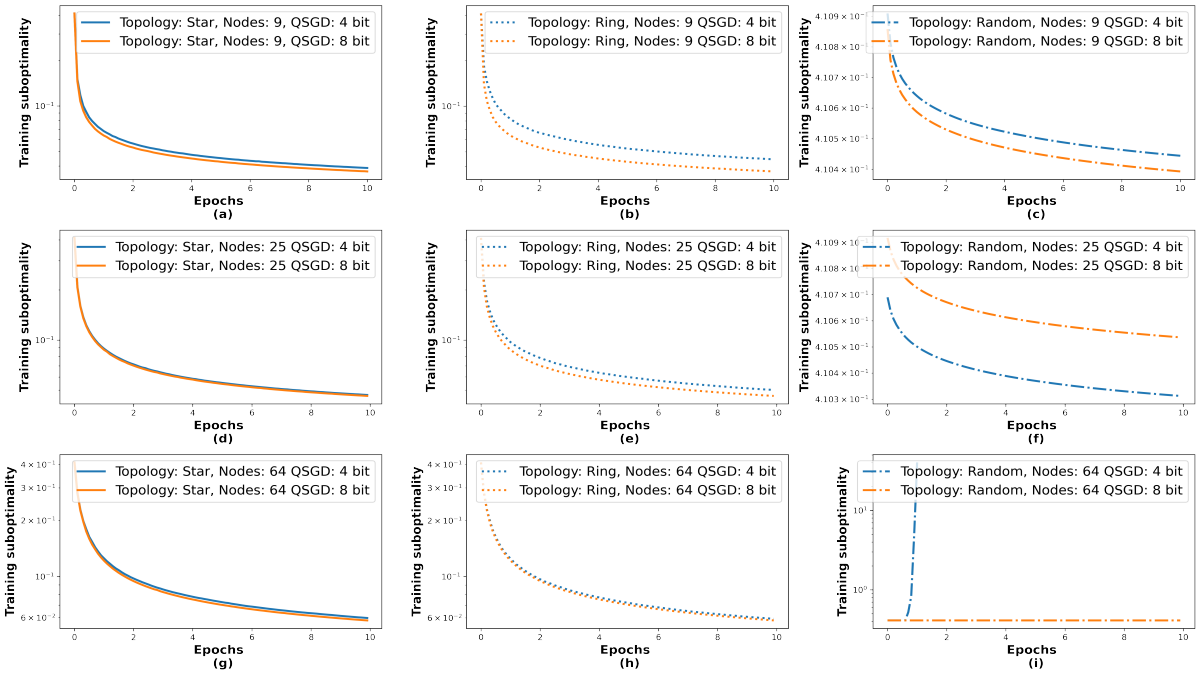

Figure 4: Comparison of DCD-PSGD algorithm for different number of quantization bits.

7.3 Part 3: Performance of CHOCO-SGD

In this part, we have tested the CHOCO SGD algorithm for different number of quantization bits and compared their performance. similar to what we did with the DCD-PSGD algorithm in Part 2. In general, we see sam ekind of trend that we saw with the DCD-PSGD algorihm. From Fig 5 we find a larger number of nodes lead to a faster convergence but a slightly greater suboptimality. Also we notice that for 2 bit quantization the suboptimality increases and hence it is not a suitable quantization level.

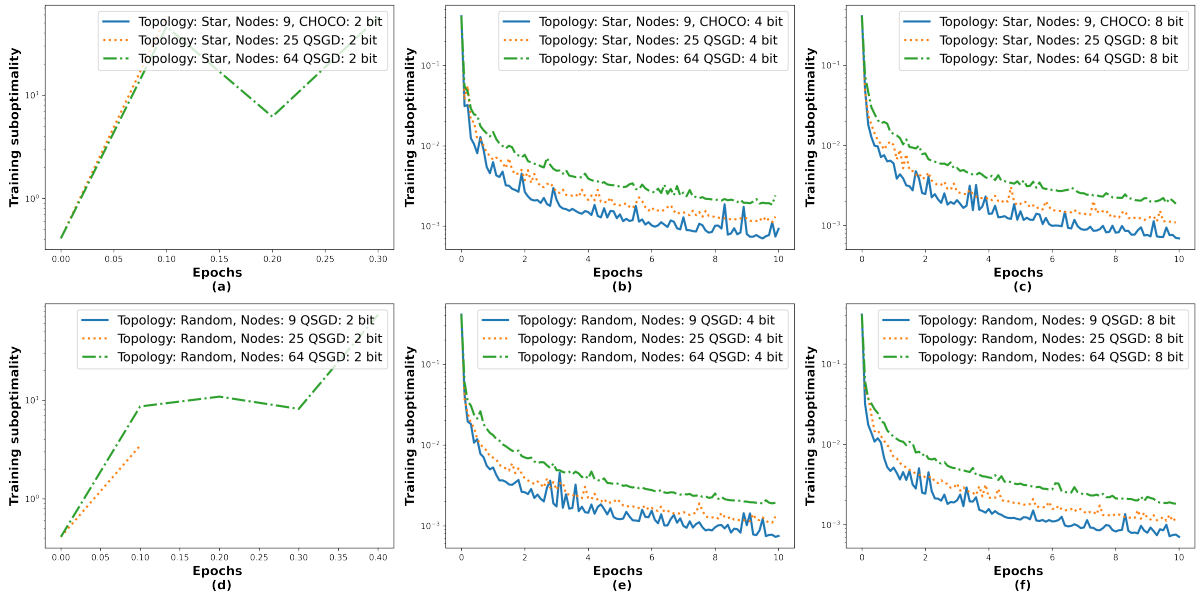

Figure 5: Comparison of CHOCO-SGD for different number of nodes.

From Fig 6 we see that the CHOCO SGD algorithm works equivalently good for both 4 and 8 quantization bits, which suggests we should use 4 bit quantization to achieve better compression.

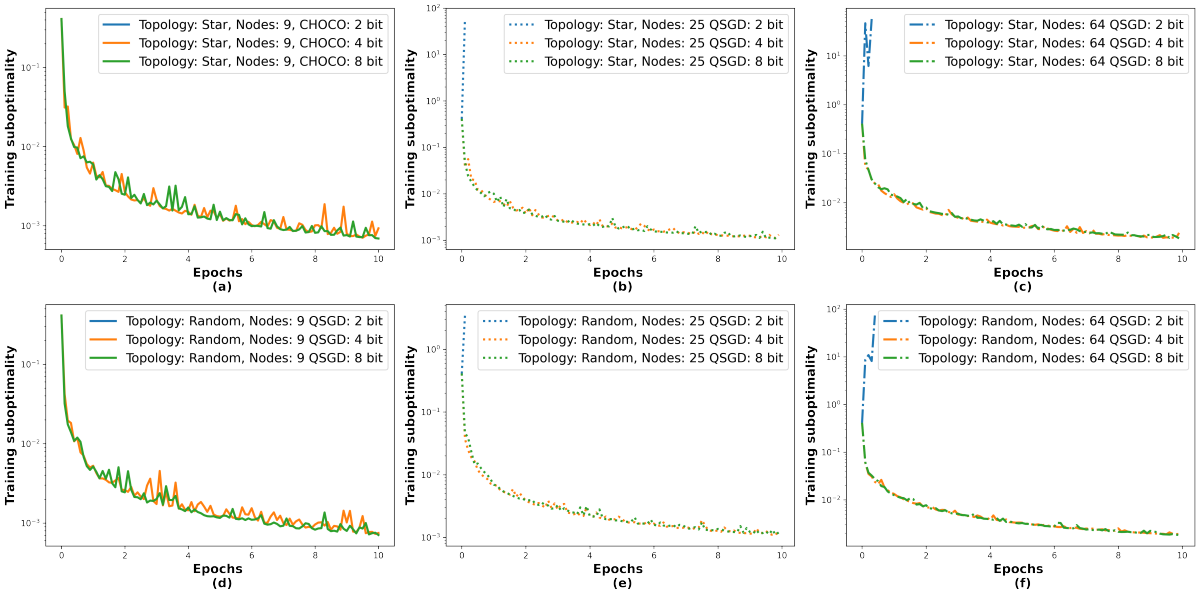

Figure 6: Comparison of CHOCO-SGD for different quantization bit.

7.4 Part 4: Performance of CHOCO and DCD with sparsification

In this last part, I have introduced sparsification in the communication process. The two sparsification methods have already been talked about above. I test both DCD-PSGD and CHOCO SGD algorithm by applying both quantization and sparsification.

There are two types of sparsification that are introduced and tested $rand_k$ and top_k. I have tested for top 200 and top 400 coordinates and found 400 coordinates to be an appropriate number for CHOCO method. The total number of coordinates are 2000.

Fig. 7 shows the comparison of the two algorithms (with the plain algorithm) for 4 bit quantization. Clearly we see that CHOCO method performs better than DCD and very close to the plain Decentralized SGD. This suggests that 4 bit quantization is suitable for the CHOCO method. Fig. 8 shows the comparison between CHOCO and CHOCO-TOP-400 with 4 bit quantization. The performance of both the algorithms for different number of nodes is almost the same which suggests that along with compressing the bits we can also just use only 400 co-ordinates rather than all 2000.

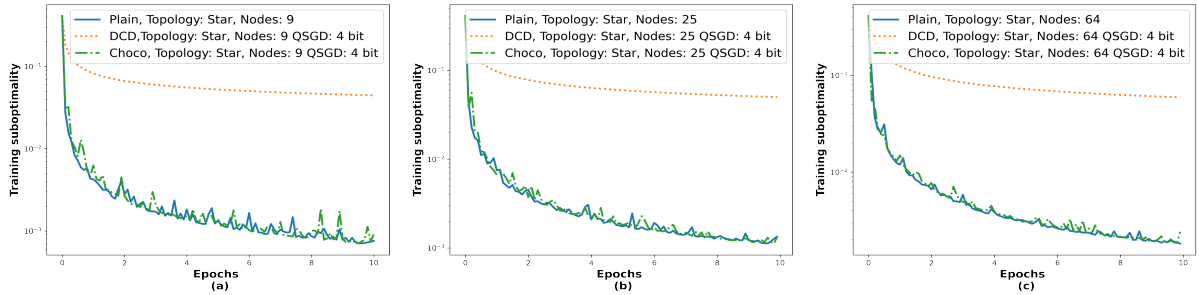

Figure 7: Comparison of Choco, DCD and Plain with 4 bit quantization

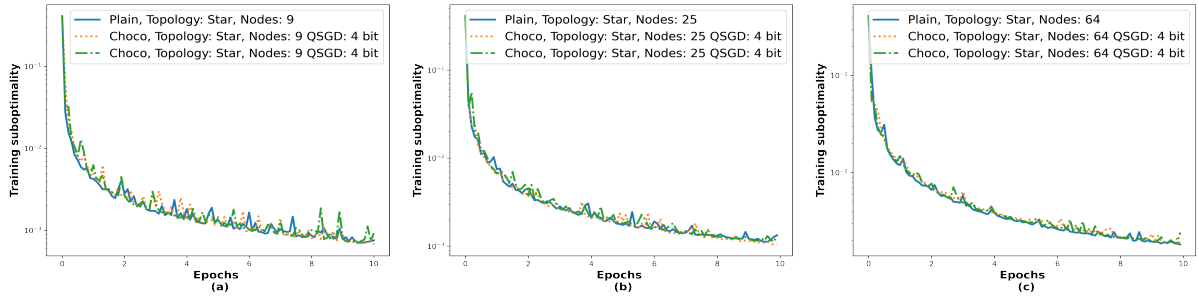

Figure 8: Comparison of Choco, Choco-Top 400 and Plain with 4 bit quantization

7.5 Part 5: Random Activation with Constant Bits

In this part we activate only a certain number of nodes to broadcast their data in order to decrease the communication cost. Fig. 9 shows the performance of the Choco-SGD Method with random activation of bits with a different probability. The training suboptimality increases as we decrease the probability of activating a particular node. A less probability means there are less number of nodes in the network that are broadcasting there data and hence the increased loss. The case where $p = 1$ is the same as the CHOCO method (full activation). When the activation probability $p = 0$, it means no node is communicating (only self communication).

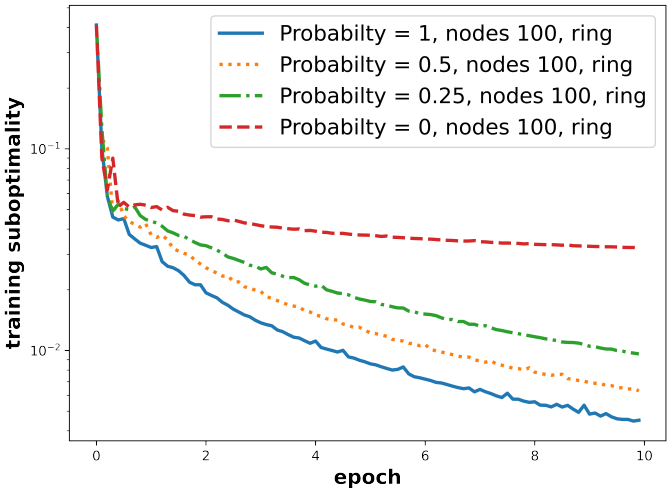

Figure 9: Comparison of new method for different p for constant number of bits

This result suggests it is better to have more number of nodes communicating with each other.

8 Conclusion

In this work we compared the performance of two of the existing algorithm to solve Decentralized Stochastic Algorithm and compared the effect of changing the number of quantization bits, topology and number of nodes on these algorithm. We saw that there is a slight effect of topology on the convergence of the algorithms. Quantization bits play a huge role in the performance of these algorithms. We saw this performance for 2, 4, and 8 bit quantization levels and found 4 to be the cut-off point. Using quantization bits lesser than 4 bits make the algorithms diverge. Finally we introduced the idea of random bit activation in the in the already existing CHOCO gossip algorithm to reduce the communication cost of the network.

References

[1] Y. Saad, "Numerical solution of large nonsymmetric eigenvalue problems," *Computer Physics Communications*, vol. 53, no. 1-3, pp. 71–90, 1989.

[2] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[3] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3478–3487.

[4] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[5] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[6] F. Fagnani and S. Zampieri, "Average consensus with packet drop communication," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 1007–1012.

[7] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "An improved glmnet for l1-regularized logistic regression," *Journal of Machine Learning Research*, vol. 13, no. 64, pp. 1999–2030, 2012. [Online]. Available: http://jmlr.org/papers/v13/yuan12a.html