Approximate Weighted *CR* Coded Matrix Multiplication

[†] **Neophytos Charalambides** [‡] Mert Pilanci [†] Alfred Hero

[†] EECS Department University of Michigan, [‡] EE Department Stanford University

June, 2021

Issues and Motivation

Introduction and Motivation

Machine Learning Today : *Curse of Dimensionality*

- Large Datasets — many samples
- Complex Datasets — large dimension
- Problems become intractable

Use distributed methods

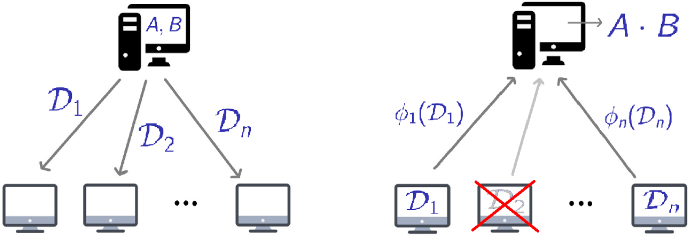
- Distribute smaller computation assignments
- Multiple servers complete various tasks

Drawbacks of Distributed Synchronous Computations

- Requires all servers to respond — communication overhead
- What if stragglers are present ?
- Stragglers — servers with delays or non-responsive

Coded Matrix Multiplication (CMM)

1. Speed up distributive computation — matrix multiplication
2. Mitigate stragglers



Multiplying A, B :

- Partition A, B and send information $\mathcal{D}_i$ to the workers
- Workers compute $\phi_i(\mathcal{D}_i)$ and send it back
- Main server recovers $A \cdot B$
- Waits for $f = n - s$ fastest workers (s stragglers, out of n workers)

Our Motivation and Approaches

Previous Methods :

- Many different coding approaches :
  - Polynomial, Short-dot, MatDot, MDS, Binary, Polar codes, etc.
- Consider *exact* recovery — high computations
- Few *approximate* schemes
- Approximate MM suffices in ML applications

Our Approach :

- Use outer-product for ϕ_i, and combine with weighting
- Leverage *CR* approximate multiplication
- Weighting results in further compression
- **1st approach** : Use *any* gradient code to devise a Weighted-CMM
- **2nd approach** : Utilize MatDot CMM

CR-Multiplication [1]

- Consider $A \in \mathbb{R}^{L\times N}$ and $B \in \mathbb{R}^{N\times M}$
- Let $A^{(j)} = j^{th}$ column of A, and $B_{(j)} = j^{th}$ row of B
- $AB = \sum_{j=1}^{N} A^{(j)} B_{(j)}$
- Sample from $\left\{\left(A^{(j)}, B_{(j)}\right)\right\}_{j=1}^{N}$ *with replacement*, with probability :

$$p_i \propto \|A^{(i)}\|_2 \cdot \|B_{(i)}\|_2$$

- For $r < N$ sampling trials, with index multiset $\mathcal{S}$:

$$AB \approx \frac{1}{r} \cdot \left(\sum_{j\in\mathcal{S}} \frac{1}{p_j} A^{(j)} B_{(j)}\right) = \sum_{j\in\mathcal{S}} \frac{A^{(j)}}{\sqrt{rp_j}} \cdot \frac{B_{(j)}}{\sqrt{rp_j}}$$

We generalize this to sampling blocks

1. Drineas et al., "*Fast MC Algorithms for Approximate Matrix Multiplication*", FOCS 2001

Block CR-Multiplication

- Partition A and B into $K = N/\tau$ block pairs :

$$A = \begin{bmatrix} \tilde{A}_1 & \cdots & \tilde{A}_K \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} \tilde{B}_1^T & \cdots & \tilde{B}_K^T \end{bmatrix}^T$$

- $\tilde{A}_i \in \mathbb{R}^{L \times \tau}$ and $\tilde{B}_i \in \mathbb{R}^{\tau \times M} \quad \Longrightarrow \quad \tilde{A}_i \tilde{B}_i$ is a rank-τ outer product
- We consider $t = r/\tau$ sampling trials, with index multiset $\bar{\mathcal{S}}$ (s.t. $|\bar{\mathcal{S}}| < K$)

$$\tilde{C} = \frac{1}{\sqrt{t}} \left[\tilde{A}_{\bar{\mathcal{S}}_1} / \sqrt{\Pi_{\bar{\mathcal{S}}_1}} \quad \cdots \quad \tilde{A}_{\bar{\mathcal{S}}_t} / \sqrt{\Pi_{\bar{\mathcal{S}}_t}} \right] \in \mathbb{R}^{L \times t\tau}$$

$$\tilde{R} = \frac{1}{\sqrt{t}} \left[\tilde{B}_{\bar{\mathcal{S}}_1}^T / \sqrt{\Pi_{\bar{\mathcal{S}}_1}} \quad \cdots \quad \tilde{B}_{\bar{\mathcal{S}}_t}^T / \sqrt{\Pi_{\bar{\mathcal{S}}_t}} \right]^T \in \mathbb{R}^{t\tau \times M}$$

- Optimal sampling distribution that minimizes the variance of $Y = \tilde{C}\tilde{R}$ is :

$$\Pi_i = \frac{\|\tilde{A}_i\|_F \|\tilde{B}_i\|_F}{\sum_{l=1}^{K} \|\tilde{A}_l\|_F \|\tilde{B}_l\|_F} \qquad \text{for } i = 1, 2, ..., K$$

Main Result

Theorem (Theorem 2.1)

The estimator $Y = \tilde{C}\tilde{R}$ is unbiased, while the sampling probabilities $\{\Pi_i\}_{i=1}^K$ minimize $\mathrm{Var}(Y)$, and $\|AB - \tilde{C}\tilde{R}\|_F^2 = O\left(\|A\|_F^2\|B\|_F^2/t\right)$.

For convenience, we can define $\mathbf{S} \in \mathbb{R}^{N\times r}$ s.t. :

$$\tilde{C} = A\cdot\mathbf{S} \text{ and } \tilde{R} = \mathbf{S}^T\cdot B \implies AB \approx A(\mathbf{S}\mathbf{S}^T)B .$$

The matrix $\mathbf{S}$ is determined by $\bar{\mathcal{S}}$ and $\{\Pi_i\}_{i=1}^K$.

Weighted *CR*-Multiplication

Idea : Only consider one copy of sampled pairs, and *weight* them accordingly.

- Sample until t *distinct* blocks are drawn
- For each $\iota \in \bar{\mathcal{S}}$, let $\mathbf{w}_\iota = \#\{\text{times } \iota \text{ is in } \bar{\mathcal{S}}\}$
  - This gives us the weight vector $\mathbf{w} \in \mathbb{N}_0^{1\times K}$
- $\mathcal{I} = \bar{\mathcal{S}} \cap \mathbb{N}_K$ the index set of $\bar{\mathcal{S}}$, i.e. $|\mathcal{I}| = t$
- $\sum_{j\in\bar{\mathcal{S}}} \tilde{A}^{(j)}\tilde{B}_{(j)} = \sum_{i\in\mathcal{I}} \mathbf{w}_i \cdot \tilde{A}^{(i)}\tilde{B}_{(i)}$
- By appropriately reweighting, we have $\mathbf{S}_\mathbf{w}$ s.t. $\underline{\tilde{C}\tilde{R} = A(\mathbf{S}_\mathbf{w}\mathbf{S}_\mathbf{w}^T)B}$

Theorem (Proposition 2.3)

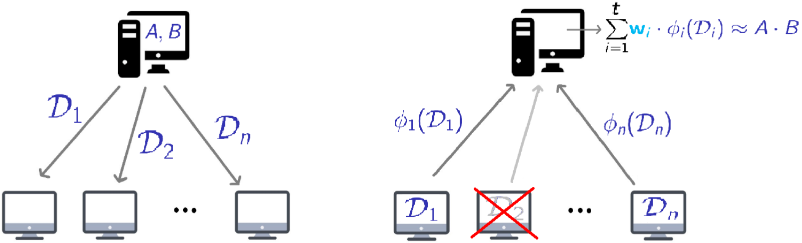
The resulting approximations from the algorithms using $\mathbf{S}$ *and* $\mathbf{S}_\mathbf{w}$*, are identical.*

Benefit : Consider $\|\mathbf{w}\|_1$ many sampled pairs, while only storing $\|\mathbf{w}\|_0$.

More succinct representation.

Weighted CMM

- Construct CMM schemes which return weighted sum of outer products
  - regardless of which workers respond, we will always have the same weighted approximate result
  - encoding $\tilde{\mathbf{B}}$, decoding $\tilde{\mathbf{a}}$



- $\mathcal{D}_i = \left(\frac{1}{\sqrt{t\Pi_i}} \cdot \tilde{A}_i, \frac{1}{\sqrt{t\Pi_i}} \cdot \tilde{B}_i\right) \quad \rightsquigarrow \quad \phi_i(\mathcal{D}_i) = \frac{1}{t\Pi_i} \cdot \tilde{A}_i^T \tilde{B}_i$
- $\mathcal{F}$ is the index set of the f responsive workers
- We provide two approaches

WCMM Schemes From Gradient Coding

- The output of a gradient code is the *sum of vectors*[2] — the partial gradients

  Condition : $\mathbf{a}_{\mathcal{F}}^T \cdot \mathbf{B} = \mathbf{1}_{1\times t}$ for all possible $\mathcal{F}$

- Let $\mathbf{X} = \begin{bmatrix} \phi_1(\mathcal{D}_1) & \cdots & \phi_t(\mathcal{D}_t) \end{bmatrix}$

- Let $\tilde{\mathbf{w}}$ be the restriction of $\mathbf{w}$ to nonzero entries

Consider any GC $(\mathbf{a}_{\mathcal{F}}, \mathbf{B})$:

- **Encoding** : $\tilde{\mathbf{B}} \coloneqq \mathbf{B} \cdot diag(\tilde{\mathbf{w}}) \otimes \mathbf{I}_L$
- **Decoding** : $\tilde{\mathbf{a}}_{\mathcal{F}}^T \coloneqq \mathbf{a}_{\mathcal{F}}^T \otimes \mathbf{I}_L$

$$\implies \quad \tilde{\mathbf{a}}_{\mathcal{F}}^T \cdot \left(\tilde{\mathbf{B}} \cdot \mathbf{X}\right) = \cdots = (\tilde{\mathbf{w}} \otimes \mathbf{I}_L) \cdot \mathbf{X} = \sum_{i=1}^{t} \tilde{\mathbf{w}}_i \cdot \phi_i(\mathcal{D}_i)$$

Theorem (Proposition 3.2)

After compressing A, B by $\rho > 1$, we can now tolerate $\grave{s} = \rho(s+1) - 1$ stragglers.

2. Tandon et al., "*Gradient Coding : Avoiding Stragglers in Distributed Learning*", ICML 2017

WCMM Scheme From MatDot CMM

- MatDot[3] is a polynomial CMM which utilizes outer products
- Let $x_1, \ldots, x_n \in \mathbb{F}_q$ distinct for $q > n$
- **Encoding** : $p_A(x) = \sum_{j=1}^{t} \tilde{A}_j x^{j-1} \qquad p_B(x) = \sum_{j=1}^{t} \tilde{B}_j x^{t-j}$
- $C(x_i) = p_A(x_i) \cdot p_B(x_i)$ computed and communicated by the i^{th} worker
- **Decoding** : Polynomial interpolation, once $2t-1$ evaluations are received
- **Weighting** : $\tilde{p}_A(x) = \sum_{j=1}^{t} \sqrt{\tilde{\mathbf{w}}_j} \cdot \tilde{A}_j x^{j-1} \qquad \tilde{p}_B(x) = \sum_{j=1}^{t} \sqrt{\tilde{\mathbf{w}}_j} \cdot \tilde{B}_j x^{t-j}$

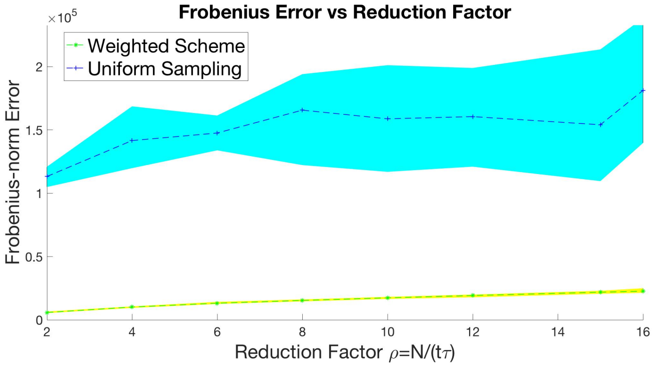
Theorem (Proposition 3.3)

Our recovery threshold drops from $2t-1$ to $2\grave{t}-1 = 2(t/\rho)-1$.

3. Fahim et al., "*On the Optimal Recovery Threshold of CMM*", Allerton 2017

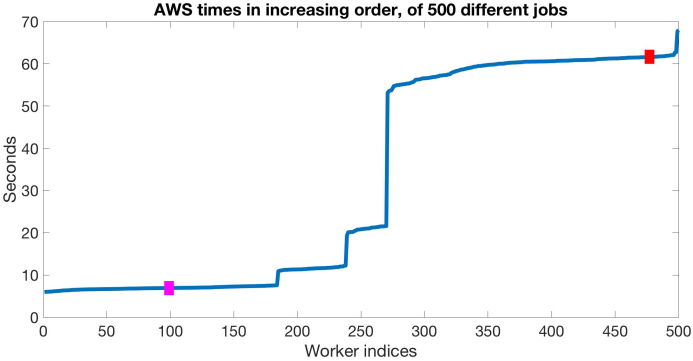
Minimum Variance of Frobenius Error

- Constructed $A \in \mathbb{R}^{260\times 9600}$, $B \in \mathbb{R}^{9600\times 280}$ with non-uniform $\{\Pi_i\}_{i=1}^K$
- $K = 280 \rightsquigarrow \tau = 20$, with $\|A\|_F^2\|B\|_F^2 = O(10^{11})$
- Considered error $\|AB - \tilde{C}\tilde{R}\|_F^2$ and varying t
- Ran the approximation 10 times for each t
- Compared it against a uniformly sampling scheme



AWS Jobs with a GC Approach

- Same set up, with $N = 10^4$, $K = 500 \rightsquigarrow \tau = 20$
- Consider AWS times [4], with $n = 500$ and $\rho = 20$
- For the same completion time :
  - Unweighted : $s = 19$
  - Weighted : $\grave{s} = 399$
- Only needed 10% of the overall time, and had relative error 8.26×10^{-7}



4. Bartan et al., "*Polar Coded Distributed Matrix Multiplication*", Allerton 2019

Thank you for your attention !

☺