Robust Gradient Descent via Moment Encoding with LDPC Codes

Extended Abstract

Raj Kumar Maity
UMass Amherst
Amherst, MA
rajkmaity@cs.umass.edu

Ankit Singh Rawat
MIT and UMass Amherst
Cambridge and Amherst, MA
asrawat@mit.edu

Arya Mazumdar
UMass Amherst
Amherst, MA
arya@cs.umass.edu

ABSTRACT

In this paper, we consider the problem of implementing large-scale gradient descent algorithms in a distributed computing setting in the presence of *straggling* processors (stragglers). To mitigate the effect of the stragglers, it has been previously proposed to encode the data with an erasure-correcting code and decode at the master server at the end. In this work, instead, we propose to encode the second-moment of the data with an iteratively decodable low density parity-check (LDPC) code. To analyze our scheme, we show that for a randomized model for stragglers the gradient descent method with the proposed moment-encoding based setup can be viewed as stochastic gradient descent. This leads to a convergence guarantee for our gradient descent algorithm. On the other hand, iterative decoding methods of LDPC codes come with very low computational overhead and the number of decoding iterations can be made to automatically adjust with the number of stragglers present. We have implemented the proposed moment-encoding method, and it outperforms the previously proposed schemes in a real distributed computing setup.

1 CODED DISTRIBUTED COMPUTATION

As a general principle, large scale distributed computing setups (e.g., [5, 18]) divide the original problem at hand into many small tasks, which are then assigned to many computing servers, namely workers. The master server collects the outcomes of local computation at the workers (potentially over multiple rounds) and computes the final result. In large scale practical systems, this process of collecting the outcomes from the workers is usually prone to unpredictable delays [4]. Such delays arise due to various reasons, including the slow-down at worker servers and the congestion present in the communication networks among the servers in the system. The workers that are not able to provide the outcome of their local computation within a reasonable deadline due to these delays are termed *stragglers*. The presence of the stragglers can significantly degrade the performance of the system. Therefore, it becomes critical to address the variability in the response times of different components of the computing setup during the design of the computational tasks.

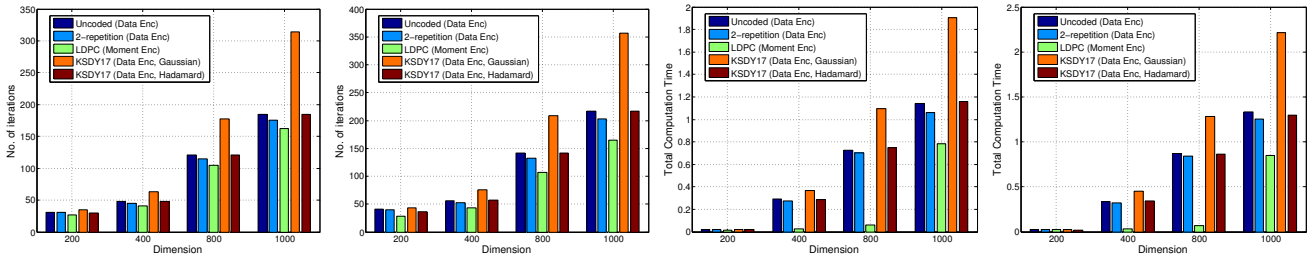
The problem of mitigating the effect of stragglers has been explored in many recent works. The *replication* schemes assign each task to multiple servers [1, 14, 16]. This ensures that the task gets completed without significant delay if at least one of the servers processing the task is non-straggler. In [9], Lee et al. explored coding theoretic ideas that went beyond the replication schemes in order to address the issue of straggling servers. They focus on linear computation, such as the product of matrix A with vector $\mathbf{x}$, and propose to encode the columns of A by a maximum distance separable (MDS) code to obtain a taller encoded matrix. The rows of the encoded matrix are distributed among workers, that are responsible for computing the inner product of the rows assigned to them with $\mathbf{x}$. The redundancy among the rows of the encoded matrix allows for computation of $A\mathbf{x}$ even if some of the servers fail to respond with the computation assigned to them. In [6], Dutta et al. further explore the problem of reliably computing a matrix-vector product with the additional requirement that the rows of the encoded matrix are sparse. This is motivated by the objective of reducing the computation at the workers and communication between the master and the workers which scales with the row-sparsity of the encoded matrix. The similar ideas for other computational tasks (e.g., matrix-matrix product and convolution between vectors) have been explored in [2, 7, 10, 17]. Another line of work that addresses the issue of minimizing the amount of communication during data shuffling using coding techniques, both with and without stragglers, is presented in [9, 11–13] and reference therein.

In [9], Lee et al. also focus on performing iterative gradient descent procedure in a distributed manner via repeatedly invoking their solution for coded computation of matrix-vector product. In this paper, we also rely on coded computation of matrix-vector product. However, we encode the second moment matrix as opposed to the plain data matrix, as done in [9]. This leads to reduced communication rounds. Furthermore, this also makes the analysis of the optimization procedure completely different from that in [9]. As another novel contributions, we utilize LDPC codes which, as discussed above, allow for both efficient decoding and control over the quality of the (approximate) gradient computed in each step of the optimization procedure. In [8], Karakus et al. also study the problem of recovering the model parameters of a linear model by solving an alternative optimization problem where both data points and their labels are encoded by the matrices with maximal (pairwise) incoherent columns. Again, our approach differs from theirs as we solve the original optimization problem itself and rely on moment encoding as opposed to data encoding.

2 METHODOLOGY AND MAIN RESULTS

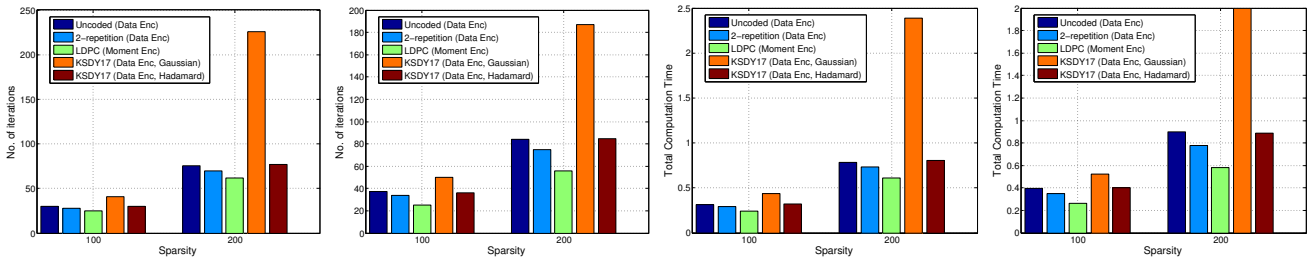
We consider two simple problems of fitting a linear model to the given data: linear regression and sparse recovery, and implement gradient descent algorithm to solve them.

Let's consider the simple linear regression task where we are interested in learning a k-dimensional vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k) \in$



(a) Number of Stragglers = 5. **(b) Number of stragglers = 10.** **(c) Number of Stragglers = 5.** **(d) Number of stragglers = 10.**

Figure 1: Total number of iterations and total computation time for solving the linear regression problem(m = 2048).



(a) Number of Stragglers = 5. **(b) Number of stragglers = 10.** **(c) Number of Stragglers = 5.** **(d) Number of stragglers = 10.**

Figure 2: Total number of iterations and computation time for solving the sparse recovery problem in an underdetermined system (k = 2000, m = 1024).

$\mathbb{R}^k$ such that the following total empirical loss function is minimized.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 = \frac{1}{2}\Big(\mathbf{y} - X\boldsymbol{\theta}\Big)^T\Big(\mathbf{y} - X\boldsymbol{\theta}\Big) = \frac{1}{2}\sum_{i=1}^{m}\Big(y_i - \mathbf{x}_i^T\boldsymbol{\theta}\Big)^2,$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_m) \in \mathbb{R}^m$ and $X = (\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_m)^T \in \mathbb{R}^{m\times k}$. Here, the gradient of the total empirical loss with respect to learning parameters $\boldsymbol{\theta}$ has the following form.

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \Big(X^TX\boldsymbol{\theta} - X^T\mathbf{y}\Big). \tag{1}$$

Note that the term $X^T\mathbf{y}$ is independent of the optimization parameter $\boldsymbol{\theta}$; as a result, this term needs to be computed only once at the beginning of the gradient descent based optimization procedure. By using the notation $M = X^TX$ and $\mathbf{b} = X^T\mathbf{y}$, we can rewrite the gradient as

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = M\boldsymbol{\theta} - \mathbf{b}. \tag{2}$$

Therefore, the t-th step of the gradient descent optimization procedure takes the following form.

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_{t-1}) = \boldsymbol{\theta}_{t-1} - \eta_t(M\boldsymbol{\theta}_{t-1} - \mathbf{b}), \tag{3}$$

where $\boldsymbol{\theta}_t$ denotes the estimate of $\boldsymbol{\theta}$ at the end of t-th step. Similarly, η_t represents the learning rates during the t-th step of the optimization procedure.

Now we distribute the task of computing matrix-vector product $M\boldsymbol{\theta}_t$ among the worker nodes. We use an LDPC code to encode the matrix $M = X^TX$. Note that, it is not necessary that the exact gradient be computed in every step by the master for the gradient descent algorithm to succeed. Moreover, the LDPC decoder has very low computation complexity and can automatically adjust to the number of the stragglers with small number of iteration of the decoding required if there are not too many stragglers present. Additionally, we can use the number of decoding iterations as a tuning parameters. Depending on the number of stragglers, we can run only those many decoding iterations that are sufficient to ensure the desirable quality of the estimate of the gradient during each gradient descent step. For a random model for stragglers, the gradient descent method with the proposed moment encoding based setup can be viewed as stochastic gradient descent (SGD) method. We can use the standard convergence analysis for SGD method to establish the convergence guarantee for our proposed solution. The details can be found in the full-version of this paper.

The sparse-recovery problem seeks to find a sparse vector $\boldsymbol{\theta}$ from samples of the form $\mathbf{y} = X\boldsymbol{\theta}$. We employ a gradient-based method similar to above in conjunction with a hard-thresholding operation [3] in every step of the optimization procedure to achieve the goal.

We also conduct a detail performance evaluation of our solution on a distributed computing cluster (Swarm2) at the University of Massachusetts Amherst [15]. The obtained performance results show that the our proposed solution requires smaller number of gradient steps to converge to the solution of the correct model parameters. The results are plotted in Figures 1 and 2. We compared our LDPC based (rate= 1/2) moment-encoding scheme with the recently proposed data encoding (with MDS/Gaussian matrices) scheme of Karakus et al. (KSDY17 in the figures) [8], as well as with uncoded and replication-based schemes (2-replication).

Acknowledgement: This research is supported by NSF CCF awards 1642658, 1642550 and 1618512.

REFERENCES

[1] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica. 2013. Effective Straggler Mitigation: Attack of the Clones. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 185–198. http://dl.acm.org/citation.cfm?id=2482626.2482645

[2] R. Bitar, P. Parag, and S. E. Rouayheb. 2017. Minimizing latency for secure distributed computing. In *Proceedings of 2017 IEEE International Symposium on Information Theory (ISIT)*. 2900–2904.

[3] Thomas Blumensath and Mike E Davies. 2009. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis* 27, 3 (2009), 265–274.

[4] J. Dean and L. A. Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

[5] J. Dean and S. Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[6] S. Dutta, V. Cadambe, and P. Grover. 2016. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Advances in Neural Information Processing Systems*. 2100–2108.

[7] S. Dutta, V. Cadambe, and P. Grover. 2017. Coded convolution for parallel and distributed computing within a deadline. *arXiv preprint arXiv:1705.03875* (2017).

[8] Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. 2017. Straggler mitigation in distributed optimization through data encoding. In *Advances in Neural Information Processing Systems*. 5440–5448.

[9] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran. 2017. Speeding Up Distributed Machine Learning Using Codes. *IEEE Transactions on Information Theory* PP, 99 (2017), 1–1. https://doi.org/10.1109/TIT.2017.2736066

[10] K. Lee, C. Suh, and K. Ramchandran. 2017. High-dimensional coded matrix multiplication. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2418–2422.

[11] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr. 2015. Coded Mapreduce. In *Proceedings of 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 964–971.

[12] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr. 2016. A unified coding framework for distributed computing with straggling servers. In *Proceedings of IEEE Globecom Workshops (GC Wkshps)*. IEEE, 1–6.

[13] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr. 2017. A fundamental tradeoff between computation and communication in distributed computing. *IEEE Transactions on Information Theory* (2017).

[14] N. B. Shah, K. Lee, and K. Ramchandran. 2016. When Do Redundant Requests Reduce Latency? *IEEE Transactions on Communications* 64, 2 (Feb 2016), 715–722.

[15] Swarm2. 2018. Swarm User Documentation. https://people.cs.umass.edu/~swarm/index.php?n=Main.NewSwarmDoc. (2018). Accessed: 2018-01-05.

[16] D. Wang, G. Joshi, and G. Wornell. 2015. Using straggler replication to reduce latency in large-scale parallel computing. *ACM SIGMETRICS Performance Evaluation Review* 43, 3 (2015), 7–11.

[17] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr. 2017. Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication. *arXiv preprint arXiv:1705.10464* (2017).

[18] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*. USENIX Association, Berkeley, CA, USA, 10–10. http://dl.acm.org/citation.cfm?id=1863103.1863113