RCD-SGD: RESOURCE-CONSTRAINED DISTRIBUTED SGD IN HETEROGENEOUS ENVIRONMENT VIA SUBMODULAR PARTITIONING

Haoze He

New York University, NY, USA

Parijat Dube

IBM Research, NY, USA

ABSTRACT

The convergence of SGD based distributed training algorithms is tied to the data distribution across workers. Standard partitioning techniques try to achieve equal-sized partitions with per-class population distribution in proportion to the total dataset. Partitions having the same overall population size or even the same number of samples per class may still have Non-IID distribution in the feature space. In heterogeneous computing environments, when devices have different computing capabilities, even-sized partitions across devices can lead to the straggler problem in distributed SGD. We develop a framework for distributed SGD in heterogeneous environments based on a novel data partitioning algorithm involving submodular optimization. Our data partitioning algorithm explicitly accounts for resource heterogeneity across workers while achieving similar class-level feature distribution and maintaining class balance. Based on this algorithm, we develop a distributed SGD framework that can accelerate existing SOTA distributed training algorithms by up to 32%.

***Index Terms*—** submodular optimization, data partitioning, deep learning training, distributed SGD

1. INTRODUCTION

Stochastic gradient descent (SGD) is the skeleton of most state-of-the-art (SOTA) machine learning algorithms. Traditional SGD was designed to be run serially at a single node. However, with the increasing size of deep learning models and dataset, too much time is required to process training in single machine[1]. Parallelism in training is inevitably necessary to deal with this magnitude of model, data, and compute requirements[2]. Distributed SGD parallelizes the training across multiple workers, through different types of parallelism (model, data, pipeline) to speed up training. While parallelism can achieve high training throughput, guaranteeing the stability and convergence of SGD is challenging in distributed training. It is a complex interplay of machine learning hyperparameters and dataset distribution that contributes to convergence of distributed machine learning training. Most of the SOTA decentralized distributed machine learning frameworks [3, 4, 5, 6, 7] randomly partition the original dataset into subsets and assign them to different workers. Their convergence analysis simply assumes the subsets in each worker are independent and identically distributed (IID). However, random partitioning cannot guarantee IID at the feature-based level, the Non-IID issue may still exists.

Submodular optimization, associated with a rich family of submodular functions that can measure the diversity of a given subset, has been widely used in machine learning in the past decade. Submodular optimization is well known for its strong capability to select a representative subset. Earlier work have tried applying submodular optimization to real-world applications including speech recognition, active learning, and computer vision [8, 9, 10, 11, 12, 13, 14, 15, 16]. To solve the Non-IID issue in distributed machine learning, [17] tries to use submodular optimization to partition the original dataset into IID subsets. General partition on any dataset ahead of training won't give additional computational costs and will lead to faster convergence. However, the submodular partition algorithm GreedMax [17] will lead to different sized subsets after partition. Different sized subsets will lead to straggler problem (the delays in waiting for the learners with large batch size) due to synchronization step in most distributed SGD. Although some recent papers try to explore the possibility of using submodular optimization to partition dataset under certain constraints [18], it cannot be applied to distributed machine learning directly due to non-uniformly sized and imbalanced classes across partitions. To solve these problems, we propose RCD-SGD, a resource-constrained distributed SGD. Our main contributions are:

1. We propose a novel algorithm to partition data for distributed SGD in heterogeneous environments. The proposed algorithm partitions the dataset across workers in proportion to their computational capabilities while achieving similarity in class-level feature distribution and maintaining class balance. Our algorithm reduces the computational complexity of earlier partitioning algorithms by a factor proportional to the number of classes in the dataset [18].

2. By partitioning the original dataset into subsets with the same number of data points, the label-balanced greedy partition algorithm addresses the straggler problem in distributed synchronous SGD.

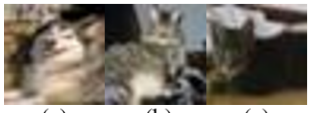
3. We evaluate the RCD-SGD algorithm using two different submodular functions and two SOTA-distributed SGD algorithms. By achieving IID partitioning of the dataset our algorithm achieves faster convergence than the SOTA baseline. With approximately local IID subsets, we reduce the communication frequency of distributed SGD and achieve up to 32% speedup in wall-clock time when compared with the SOTA algorithms [19]. The final model also achieves slightly better loss and improves the final accuracy by 1.1%.

4. The proposed resource-constrained distributed SGD is a general algorithm that is extendable to most distributed SGD SOTA algorithms. Using any algorithm as a baseline, resource-constrained distributed SGD can achieve faster convergence and shorter wall-clock training time.

2. PROPOSED METHOD

The convergence of distributed training is tied to the data distribution across workers. For efficient distributed training, the data partitions at different workers should have similar data distribution. A simple random partitioning in equal-sized partitions may not preserve class-level distribution across partitions. Class-level random partitioning ensure that the number of samples of different classes in any partition is in the same proportion as in the original dataset. However, this may still not guarantee that the feature distribution of a class is similar across different partitions.

Some recent works involve the use of submodular functions for data partitioning for efficient distributed machine learning. In [17] using a greedy algorithm involving the use of submodular functions, the dataset was partitioned into subsets with IID features. However, their greedy algorithm has three major problems:



(a) (b) (c)

Fig. 1. The similarity between dog a and cat b is higher than similarity between b and c. Similarity is calculated using cosine similarity and aussian kernel with L2 distance

Non-uniform sized partitions The algorithm can lead to different sized partitions. This will lead to a straggler problem in any distributed training algorithms with a synchronization barrier.

Class imbalance across partitions The algorithm can lead to a different number of samples per class across the partitions. This imbalance can lead to lower performance of the aggregated model. The reason why this can happen is that if there are two classes with an overlap in feature space (e.g., Figure 1), then since the greedy algorithm is only maintaining similar feature distribution across partitions, we can have uneven number of samples from these two classes in different partitions.

High computational complexity: With n samples in the dataset, the greedy algorithm requires $O(n^2)$ computations which can be prohibitive for large datasets.

The approach was generalized in [18] to perform constrained submodular partitioning. The experiment demonstrated that the average performance of models trained individually on different subsets is better than random partitioning. However, this was never demonstrated in distributed training setting with communication between workers. In addition, it cannot handle heterogeneous environments, the issue of class imbalance and high complexity remains.

2.1. Resources-Constrained submodular partitioning

Algorithm 1 Ratio-Constrained submodular partitioning for distributed SGD in heterogeneous environment

Initialization: submodular function f, ground set V , number of blocks N, number of classes L
Set the ratio of constraint according to the computational performance of different workers $r_1, r_2, ..., r_N$
Let $A_1 = A_2 = ... = A_N = \emptyset$
Split the ground set V into N sets $V_1, V_2, ..., V_L$ according to class labels.
for $V_l = V_1, V_2, ..., V_L$ **do**
  Constraint $C_{l,j} = \frac{|V_l| r_j}{\sum_{j=1}^{N} r_j}, j \in \{1, \ldots, N\}$
  Let $A_1^l = A_2^l = ... = A_N^l = \emptyset, J = [N], R = V_l$
  while $R \neq \emptyset$ and $J \neq \emptyset$ **do**
    $j^* \in \text{argmin}_{j \in J} f(A_j^l)$
    if $\exists v \in R$ s.t. $A_{j^*}^n \cup \{v\} \in C_{l,j}$ **then**
      $v^* := \text{GreedyStep}(R, C_{l,j}, A_{j^*}^l)$
      $A_{j*}^l := A_{j^*}^l \cup \{v^*\}$, R := R $\setminus \{v^*\}$
    else
      Let $J = J \setminus j^*$
    end if
  end while
  Joint subset: $A_1 = A_1 \cup A_1^l, ..., A_N = A_N \cup A_N^l$
end for
Output:$(A_1, A_2, A_3, ..., A_N)$

Algorithm 1 is our proposed algorithm for ratio-constrained submodular partitioning. In heterogeneous environments, there can be a wide gap in the compute capabilities of different workers. To solve this problem, we set different constraints in the submodular optimization algorithm according to compute performance of workers to ensure that the number of samples in a subset is proportional to the performance of the worker training with that subset. Besides, instead of doing

partitioning on the whole dataset, we do class-level partitioning of the dataset. By using consistent constrained across all classes for a certain worker, we can get balanced subclasses. Since the submodular optimization greedy algorithm requires $O(n^2)$ computations, our algorithm will reduce the computation complexity from $O(n^2)$ to $O(\frac{n^2}{L})$ in L classes dataset thereby achieving $100\times$ speedup when partitioning cifar100.

We formulate the general resources-constrained submodular partitioning optimization task as follows: partition datasets into N groups such that each group contains similar, sufficiently, and approximately-IID strong prediction power. The dataset of class l: $\mathbf{V}_l = \{\mathbf{X}, \boldsymbol{y}\}$ is given, where $\mathbf{X} \in \mathbf{R}^{N\times d}$ and $\boldsymbol{y} \in \{0,1\}^N$. Set the constraint according to the computational capability of different workers: $C_{l,1}, C_{l,2}, ..., C_{l,N}$ so that $\sum_{i=1}^{N} C_{l,i} = N$. Define a partition of V_l as $\{A_1^l, A_2^l, \ldots, A_N^l\}$ where $A_n^l \in V_l$ for $n = 1, 2, \ldots, N$ such that

$$A_j^l \cap A_i^l = \emptyset \quad \forall i, j \in \{1, 2, \ldots, N\}, i \neq j$$

$$\bigcup_{n=1}^{k} A_n^l = V_l, S_n \in R^{C_{l,n}\times d}$$

The proposed Ratio-Constrained submodular partitioning algorithm (algorithm 1) addresses the partition task. The $GreedyStep(R, C_l, A_{j^*}^l)$ in algorithm 1 utilizes the submodular function to measure the value. The GreedyStep is defined as:

$$v^* = \arg\max_{i \in V_l \setminus \hat{V}_l} f\left(A_j^l \cup \{v^*\}\right)$$

where $\bigcup_{n=1}^{k} A_n^l = \hat{V}_l \subset V_l$. After partition the subsets will finally achieve: $\bigcup_{n=1}^{k} A_n^l = V_l$. We define a discrete set function $f: \mathbf{V} \to \mathbf{R}$ as a submodular function if

$$f\left(\hat{A}_j^l \cup \{v^*\}\right) - f\left(\hat{A}_j^l\right) \geq f\left(A_j^l \cup \{v^*\}\right) - f\left(A_j^l\right)$$

where $\hat{A}_j^l \subset A_j^l, \forall j \in \{V_l \setminus \cup_{n=1}^{k} A_j^l\}$. A submodular function f must be monotone non-decreasing:

$$f\left(A_j^l \cup \{v^*\}\right) - f\left(A_j^l\right) \geq 0$$

2.2. Distributed SGD

We next propose a synchronous distributed training algorithm using our proposed partitioning algorithm. The main steps are depicted in Figure 2. Partitioning the data as Algorithm 1 ensures that each worker gets total (and per class) samples proportional to their processing speed. This will achieve approximate IID partitioning across workers, both at the feature level and label level. After partitioning the data, we perform synchronous distributed training with several rounds of local SGD before each synchronization step. This will not be detrimental to our convergence as we are ensuring IID partitions.

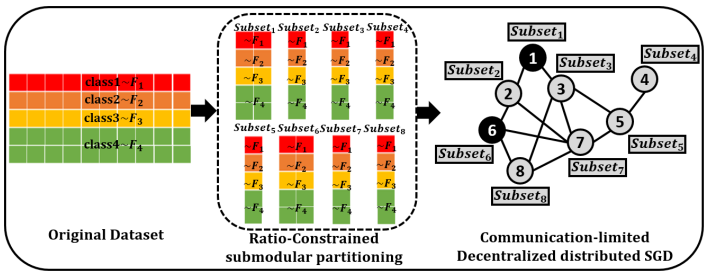


Fig. 2. RCD-SGD performs ratio-constrained partitioning of datasets, where the number of samples per class is proportional to the compute capabilities of the workers while maintaining per class feature distribution across partitions. When training multiple epochs of local SGD at workers leads to faster convergence with reduced communication overhead.

Algorithm 2 Decentralized distributed SGD algorithm

Initialization:initialize local models $\{x_0^i\}_{i=1}^N$ with the same initialization, the ratio of computational performance for different workers is $r_1, r_2, ..., r_N$, learning rate γ, batch size $B\frac{r_1}{r_i}$, communication frequency F, weight matrix W, and the total number of iterations K. Import the local subset ξ_i from subsets $\{A_1, A_2, ..., A_N\}$, which partitioned by Ratio-Constrained submodular partitioning algorithm
while $k = 0, 1, 2, ...K-1$ **do**
  Compute the local stochastic gradient $\nabla F_i(x_{k,i}; \xi_{k,i})$ on all nodes
  if k $\%$ F == 0 **then**
    Compute the neighborhood weighted average by fetching neighbor models: $\hat{x}_{k,i} = \sum_{j=1}^{N} W_{ij} x_{k,j}^b$
  else
    $\hat{x}_{k,i} = x_{k,i}$
  end if
  Update local model: $x_{k+1,i} = \hat{x}_{k,i} - \gamma \nabla F_i(x_{k,i}; \xi_{k,i})$
end while
Output: Average of all workers $\frac{1}{N}\sum_{i=1}^{N} x_{K-1,i}$

Our training algorithm is a modification of Distributed-Parallel SGD (D-PSGD) [17] where the data partitions can be Non-IID due to random partitioning. The Non-IIDness can be in feature and/or label space. Thus convergence can be poor with D-PSGD, especially for datasets with an overlap in feature space across classes. Further D-PSGD has a communication barrier after every step of local training and hence takes a longer time to converge.

During training, the communication between workers happens over a computational graph where the nodes represent the workers and the edges denote the connection between the workers. Thus ith, jth component of W, W_{ij}, is non-zero if and only if node i and node j are connected. The workers only communicate after F iterations of local SGD. Since the

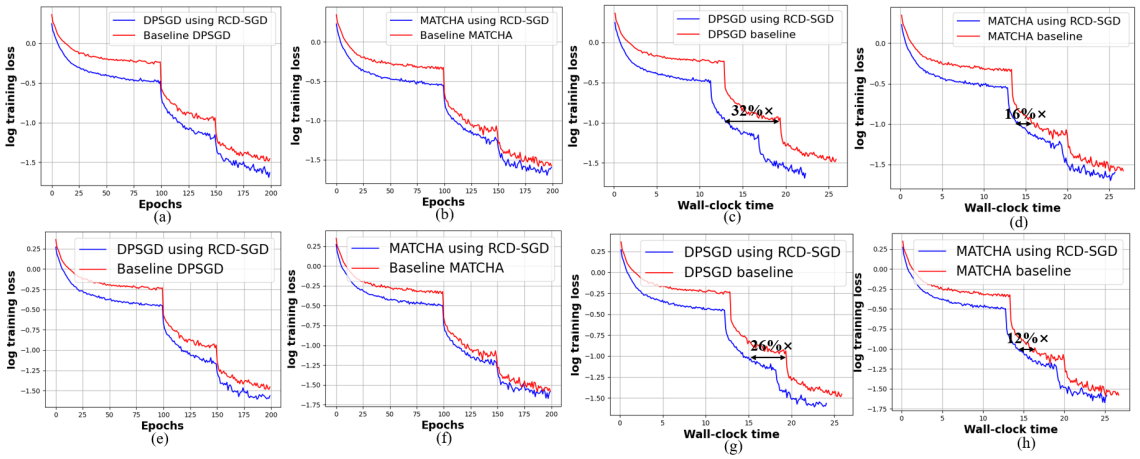


Fig. 3. RCD-SGD use facility location in $greedystep$: (a, b, c, d); use graph cut $greedystep$: (e, f, g, h)

batch size and the dataset shard per worker is proportional to its processing speed, our partitioning ensures that all the workers take approximately same amount of time to finish F iterations locally thereby reducing the straggler problem with synchronous SGD.

3. EXPERIMENTS

In Fig. 3, we report the results of RCD-SGD using D-PSGD and MATCHA as baselines. Performance of RCD-SGD using facility location and graph cut as submodular functions are included. Results are compared with baseline D-PSGD and MATCHA. Experiments use the following setting:

Models, dataset, and compared algorithms: The performance of all algorithms is evaluated on image classification task using CIFAR-10 ($|V|$ = 500000) [20]. We implement RCD-SGD as modification of D-PSGD and MATCHA with a communication budget $c_b = 0.5$. In MATCHA, each worker can communicate less frequently by setting a communication budget. The RCD-SGD implementations are compared with D-PSGD and MATCHA using random partitioning.

Submodular functions: We use facility location, i.e., $f(A_n^l) = \sum_{v \in V_l} max_{v' \in A_n^l} sim(v, v')$ and Graph Cut, i.e., $f(A_n^l) = \sum_{v \in V_l \setminus A_n^l} \sum_{v' \in A_n^l} sim(v, v')$ in $Greedystep$ (algorithm 1) separately to run the experiments. $sim(v, v')$ is the similarity between (v, v'). We use a Gaussian kernel with L2 distance to measure the similarity.

Implementations and Machines: All algorithms are trained for a sufficiently long time until convergence or onset of over-fitting. The learning rate is fine-tuned for the D-PSGD baseline and then used for all other algorithms. We set the initial learning rate as 0.8 and it decays by 10 after 100 and 150 epochs. The batch size per worker node is 64. RCD-SGD uses $F = 2$ and reduces the communication frequency to 50%. All the implementations are compiled with PyTorch and OpenMPI within mpi4py and rtx8000 GPUs as workers. We conduct experiments on a HPC cluster with 100Gbit/s Infini-band network.

Observations: For all cases, we see that RCD-SGD using either facility location or graph cut as submodular functions significantly outperform the baselines. Both the convergence rate versus epoch and convergence speed with respect to wall-clock time are improved. Up to 32% wall-clock time is saved (Fig. 3 c). Note that the training time is measured till the time the log of training loss reaches -1. Since MATCHA communicates less than DPSGD and RCD-SGD saves communication time by proportion, RCD-SGD using MATCHA as the baseline saves less communication time than RCD-SGD using DPSGD as the baseline. The performance of RCD-SGD with facility location is slightly better than the performance of RCD-SGD with graph cut. The final loss and test accuracy as shown in Table 1 are averaged over ten experiments.

Algorithms	Test Accuracy	with RCD-SGD
D-PSGD	0.925	0.937
MATCHA	0.931	0.939

Table 1. Averaged test accuracy on CIFAR-10

4. CONCLUSION

Distributed training in heterogeneous clusters requires efficient data partitioning for faster convergence. RCD-SGD achieves IID partitioning with similar per-class feature distribution across workers having different compute capabilities. The training can be performed with increased epochs of local training leading to reduced synchronization overhead. We are exploring use of other submodular functions and the sensitivity of distributed SGD convergence on their choice.

5. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al., "Efficient large-scale language model training on gpu clusters using megatron-lm," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[3] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *2019 Sixth Indian Control Conference (ICC)*. IEEE, 2019, pp. 299–300.

[5] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome, "Gossip training for deep learning," *arXiv preprint arXiv:1611.09726*, 2016.

[6] Peter H Jin, Qiaochu Yuan, Forrest Iandola, and Kurt Keutzer, "How to scale distributed deep learning?," *arXiv preprint arXiv:1611.04581*, 2016.

[7] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[8] Stefanie Jegelka and Jeff Bilmes, "Submodularity beyond submodular energies: coupling edges in graph cuts," in *CVPR 2011*. IEEE, 2011, pp. 1897–1904.

[9] Andreas Krause, Ajit Singh, and Carlos Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies.," *Journal of Machine Learning Research*, vol. 9, no. 2, 2008.

[10] Kiyohito Nagano, Yoshinobu Kawahara, and Satoru Iwata, "Minimum average cost clustering," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[11] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta, "Robust submodular observation selection.," *Journal of Machine Learning Research*, vol. 9, no. 12, 2008.

[12] Yuzong Liu, Kai Wei, Katrin Kirchhoff, Yisong Song, and Jeff Bilmes, "Submodular feature selection for high-dimensional acoustic score spaces," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7184–7188.

[13] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes, "Using document summarization techniques for speech data subset selection," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 721–726.

[14] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes, "Submodular subset selection for large-scale speech training data," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3311–3315.

[15] Kai Wei, Rishabh Iyer, and Jeff Bilmes, "Submodularity in data subset selection and active learning," in *International conference on machine learning*. PMLR, 2015, pp. 1954–1963.

[16] Jingjing Zheng, Zhuolin Jiang, Rama Chellappa, and Jonathon P Phillips, "Submodular attribute selection for action recognition in video," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[17] Kai Wei, Rishabh Iyer, Shengjie Wang, Wenruo Bai, and Jeff Bilmes, "How to intelligently distribute training data to multiple compute nodes: Distributed machine learning via submodular partitioning," in *Neural Information Processing Society (NIPS) Workshop, Montreal, Canada*, 2015.

[18] Shengjie Wang, Tianyi Zhou, Chandrashekhar Lavania, and Jeff A Bilmes, "Constrained robust submodular partitioning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2721–2732, 2021.

[19] Sebastian U Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[20] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.