Reliable Distributed Clustering with Redundant Data Assignment

Venkata Gandikota
College of Information and Computer Sciences
University of Massachusetts, Amherst, MA
gandikota.venkata@gmail.com

Arya Mazumdar
College of Information and Computer Sciences
University of Massachusetts, Amherst, MA
arya@cs.umass.edu

Ankit Singh Rawat
Google Research
New York, NY
ankitsrawat@google.com

***Abstract*—In this paper, we present distributed generalized clustering algorithms that can handle large scale data across multiple machines in spite of straggling or unreliable machines. We propose a novel data assignment scheme that enables us to obtain global information about the entire data even when some machines fail to respond with the results of the assigned local computations. The assignment scheme leads to distributed algorithms with good approximation guarantees for a variety of clustering and dimensionality reduction problems.**

I. Introduction

Clustering is one of the most basic unsupervised learning tools developed to infer informative patterns in data. Given a set of data points in $\mathbb{R}^d$, the goal in clustering problems is to find a smaller number of points, namely cluster centers, that form a good representation of the entire dataset. The quality of the clusters is usually measured using a cost function, which is the sum of the distances of the individual points to their closest cluster center.

With the constantly growing size of datasets in various domains, centralized clustering algorithms are no longer desirable and/or feasible, which highlights a need to design efficient distributed algorithms for the clustering task. In a distributed setting, we assume that a collection of data points P is too large to fit in the memory of a single server. Therefore, we employ a setup with s *compute nodes* and one *coordinator server*. In this setup, the most natural approach is to partition the data point in P into s subsets $\{P_1, \ldots, P_s\} \subset P$ and assign each of these parts to a different compute node. These nodes then perform local computation on the data points assigned to them and communicate their results to the coordinator. The coordinator then combines all the local computation received from the compute nodes and outputs the final clustering. Here, we note that the overall computation of clustering may potentially involve multiple rounds of communications between the compute nodes and the coordinator. This natural approach for distributed clustering has received significant attention from the research community. Interestingly, it is possible to obtain a clustering in the distributed setup that has its cost bounded by a constant multiple of the cost of clustering achievable by a centralized algorithm (see [1]–[3] and references therein).

In this paper, we aim to address the issue of stragglers that arises in the context of large scale distributed computing systems, especially the ones running on the so-called cloud. The stragglers correspond to those compute nodes that take significantly more time than expected (or fail) to complete and deliver the computation assigned to them. Various system-related issues lead to this behavior, including unexpected background tasks such as software updates being performed on the compute nodes, power outages, and congested communication networks in the computing setup. Some simple approaches used to handle stragglers include ignoring them and relying on asynchronous methods. The loss of information arising due to the straggler nodes can be traded for efficiency for specific tasks such as computing distributed gradients [4]–[6]. However, with the existing methods for unsupervised learning tasks such as clustering or dimensionality reduction, ignoring the stragglers can lead to extremely poor quality solutions.

Alternatively, one can distribute data to the compute nodes in a redundant manner such that the information obtained from the non-straggler nodes is sufficient to compute the desired function on the entire dataset. Following this approach, multiple coding based solutions (mainly focusing on the linear computation and first-order methods for optimization) have been recently proposed (e.g., see [4]–[10]).

This paper focuses on the relatively unexplored area of designing straggler-resilient unsupervised learning methods for distributed computing setups. We study a general class of distributed clustering problems in the presence of stragglers. In particular, we consider the k-medians clustering problem (cf. Section III-B) and the *subspace clustering* problem (cf. Section III-C). Note that the subspace clustering problem covers both the k-means and the principal component analysis (PCA) problems as special cases. The proposed Straggler-resilient clustering methods in this paper rely on a redundant data distribution scheme that allows us to compute provably good-quality cluster centers even in the presence of a large number of straggling nodes (cf. Section III-A and III-D). In Section IV, we empirically evaluate our proposed solution for k-median clustering and demonstrate its utility.

II. Background

In this section, we first provide the necessary background on the clustering problems studied in the paper. We then formally state the main objective of this work.

A. Distributed clustering

Given a dataset with n points $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subseteq \mathbb{R}^d$, distributed among s compute nodes, the goal in distributed clustering problems is to find a set of k cluster centers $C = \{\mathbf{c_1}, \ldots, \mathbf{c_k}\} \subset \mathbb{R}^d$ that closely represent the entire dataset. The quality of these centers (and the associated clusters) is usually measured by a cost function $\text{cost}(P, C)$. Two of the most prevalent cost functions for clustering are the k-median and the k-means functions, which are defined as follows.

1) k**-median**: $\text{cost}(P, C) = \sum_{i\in[n]} d(\mathbf{p}_i, C)$,
2) k**-means**: $\text{cost}(P, C) = \sum_{i\in[n]} d^2(\mathbf{p}_i, C)$,

where, $d(\mathbf{x}, \mathbf{y})$ denotes the Euclidean distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and $d(\mathbf{x}, C) := \min_{\mathbf{c}\in C} d(\mathbf{x}, \mathbf{c})$. We denote the cluster associated with the center $\mathbf{c} \in C$ by $\text{cluster}(\mathbf{c}, P) := \{\mathbf{x} \in P : \mathbf{c} = \arg\min_{\mathbf{c}'\in C} d(\mathbf{x}, \mathbf{c}')\}$. For any $\alpha > 1$, the set of cluster centers C, is called an α-approximate solution to the clustering problem if the cost of clustering P with C, $\text{cost}(P, C)$, is at most α times the optimal (minimum) clustering cost with k-centers.

In certain applications, the dataset P is *weighted* with an associated non-negative weight function $w : P \to \mathbb{R}$. The k-means cost for such a weighted dataset (P, w) is defined as $\text{cost}(P, C, w) = \sum_{i\in[n]} w(\mathbf{p}_i)\ d^2(\mathbf{p}_i, C)$. The k-median cost for (P, w) is analogously defined.

We also consider a general class of ℓ_2-error fitting problems known as the (r, k)-subspace clustering problem.

Definition 1 ((r, k)-subspace clustering)**.** *Given a dataset $P \subset \mathbb{R}^d$ find a set of k-subspaces (linear or affine) $\mathcal{L} = \{L_i\}_{i=1}^k$, each of dimension r, that minimizes $\text{cost}(P, \mathcal{L}) := \sum_{i=1}^n \min_{L\in\mathcal{L}} d^2(\mathbf{p}_i, L)$.*

Note that for $r = 0$, this is exactly the k-means problem described above. Another special case, when $k = 1$, is known as principal component analysis (PCA). If we consider the matrix $M \in \mathbb{R}^{n\times d}$, with the data points in P as its rows, it is well-known that the desired subspace is spanned by the top r-right singular vectors of M.

B. Coresets and clustering

In a distributed computing setup, where i-th compute node stores a partial dataset $P_i \subset P$, one way to perform distributed clustering is to have each node communicate a summary of its local data to the coordinator. An approximate solution to the clustering problem can then be computed from the combined summary received from all the compute nodes. This summary, called a *coreset*, is essentially a weighted set of points that approximately represents the original set of points in P.

Definition 2 (ϵ-coreset)**.** *For $0 < \epsilon < \frac{1}{3}$, an ϵ-coreset for a dataset P with respect to a cost function $\text{cost}(\cdot, \cdot)$ is a weighted dataset S with an associated weight function $w : S \to \mathbb{R}$ such that, for any set of centers C, we have*

$$(1 - \epsilon)\ \text{cost}(P, C) \le \text{cost}(S, C, w) \le (1 + \epsilon)\ \text{cost}(P, C).$$

The next results shows the utility of a coreset for clustering.

Theorem 1 ([11])**.** *Let (S, w) be an ϵ-coreset for a dataset P with respect to the cost function $\text{cost}(\cdot, \cdot)$. Any α-approximate solution to the clustering problem on input S, is an $\alpha(1+3\epsilon)$-approximate solution to the clustering problem on P.*

C. Straggler-resilient distributed clustering

The main objective of this paper is to design the distributed clustering methods that are robust to the presence of straggling nodes. Since the straggling nodes are unable to communicate the information about their local data, the distributed clustering method may miss valuable structures in the dataset resulting from this information loss. This can potentially lead to clustering solutions with poor quality (as verified in Section IV).

Given the prevalence of the stragglers in modern distributed computing systems, it is natural to desire clustering methods that generate provably good clustering solutions despite the presence of stragglers. Let OPT be the cost of the best clustering solution for the underlying dataset. In this paper, we explore the following question: *Given a dataset P and distributed computing setup with s compute nodes where at most t nodes may behave as stragglers, can we design a clustering method that generates a solution with the cost at most $c \cdot \text{OPT}$, for a small approximation factor $c \ge 1$?*

In this paper, we affirmatively answer this question for the k-median clustering and the (r, k)-subspace clustering. Our proposed solutions add on to the growing literature on straggler mitigation via coded computation, which has primarily focused on the supervised learning tasks so far.

III. Main Results

We propose to systematically modify the initial data assignment to the compute nodes in order to mitigate the effect of stragglers. In particular, we employ redundancy in the assignment process and map every vector in the dataset P to multiple compute nodes. This way each vector affects the local computation performed at multiple compute nodes, which allows us to obtain final clusters at the coordinator server by taking into account the contribution of most of the vectors in P even when some of the compute nodes behave as stragglers.

We first introduce the assignment schemes with *straggler-resilience property*. This property enables us to combine local computations from non-straggling compute nodes at the coordinator while preserving most of the relevant information present in the dataset P for the underlying clustering task. Subsequently, we utilize such an assignment scheme to obtain good-quality solutions to the k-medians and the (r, k)-subspace clustering problem in Section III-B and Section III-C, respectively. Finally, in Section III-D, we propose a randomized construction of an assignment scheme with the desired straggler-resilience property.

A. Straggler-resilient data assignment

Let the compute nodes in the system be indexed by the set $[s] := \{1, \ldots, s\}$. Furthermore let $\mathbf{p} \in P$ be assigned to the compute nodes indexed by the set $\mathcal{A}_\mathbf{p} \subset [s]$. We can alternatively represent the overall data assignment $\mathcal{A} = \{\mathcal{A}_\mathbf{p}\}_{\mathbf{p}\in P}$

by an *assignment matrix* $A \in \{0,1\}^{s\times n}$, where the n columns and the s rows of A are associated with distinct points in P and distinct compute nodes, respectively. In particular, the j-th column of A, which corresponds to the data point $\mathbf{p}_j$, is an indicator of $\mathcal{A}_{\mathbf{p}_j}$, i.e., $A_{i,j} = 1$ if and only if $i \in \mathcal{A}_{\mathbf{p}_j}$. For any $i \in [s]$, we denote the set of data points allocated to the i-th compute node by $P_i = \{\mathbf{p} \in P : i \in \mathcal{A}_{\mathbf{p}}\}$.

Let $\mathcal{R} \subset [s]$ denote the set of non-straggling compute nodes. We assume that $|\mathcal{R}| \geq s - t$, where $t < s$ denotes an upper bound on the number of stragglers in the system. Let $A_{\mathcal{R}} \in \{0,1\}^{|\mathcal{R}|\times n}$ denote the submatrix of A with only the rows corresponding to the non-straggling compute nodes (indexed by $\mathcal{R}$). For any such set of non-stragglers $\mathcal{R}$, we require that the assignment matrix A satisfies the following property.

Property 1 (Straggler-resilience property)**.** *Let $\delta > 0$ be a given constant. For every $\mathcal{R} \subset [s]$ with $|\mathcal{R}| \geq s-t$, there exists a recovery vector, $\mathbf{b} = (b_1, \ldots, b_{|\mathcal{R}|}) \in \mathbb{R}^{|\mathcal{R}|}, b_i \geq 0 \forall i \in [n]$, such that*

$$\mathbf{b}^T A_{\mathcal{R}} = (a_1, \ldots, a_n), \textit{where}, 1 \leq a_j \leq 1 + \delta \ \forall j \in [n]. \quad (1)$$

Remark 2. *The straggler-resilience property is closely related to the gradient coding introduced in [5]. However, two key points distinguish our work from the gradient coding work. First, the recovery vector $\mathbf{b}$ is restricted to have only non-negative coordinates. Second, and more importantly, the utilization of the redundant data assignment in this work (cf. Lemma 1) differs from that of gradient coding in [5] where gradient coding is used to recover the full-gradient.*

The following result, which is based on the combinatorial characterization for the assignment scheme enforced by Property 1, enables us to combine the information received from non-stragglers to generate close to optimal clustering solutions.

Lemma 1. *Let $P \subset \mathbb{R}^d$ be a dataset distributed across s compute nodes using an assignment matrix A that satisfies Property 1. Let $\mathcal{R}$ denote the set of non-straggler nodes. For any $\delta > 0$, let $\mathbf{b} \in \mathbb{R}^{|\mathcal{R}|}$ be the recovery vector corresponding to $\mathcal{R}$. Then, for any set of k centers $C \subset \mathbb{R}^d$, and any weight function $w : P \to \mathbb{R}$,*

$$\text{cost}(P, C, w) \leq \sum_{i\in\mathcal{R}} b_i \, \text{cost}(P_i, C, w) \leq (1+\delta)\text{cost}(P, C, w).$$

Equipped with Lemma 1, we are now in the position to describe our solutions for the straggler-resilient clustering.

B. Straggler-resilient distributed k-median

We distributed the dataset P among the s compute nodes using an assignment matrix that satisfies Property 1. Each compute node sends a set of (weighted) k-medians centers of their local datasets which when combined at the coordinator gives a summary for the entire dataset. Thus, the weighted k-median clustering on this summary at the coordinator gives a good quality clustering solution for the entire dataset P. Algorithm 1 provides a detailed description of this approach.

Algorithm 1 Straggler-resilient distributed k-medians

1: Input: A collection of n vectors $P \subset \mathbb{R}^d$.
2: Allocate P to s workers according to A with Property 1.
3: For each $i \in [s]$, construct Y_i, the k-median centers in P_j. Define function $w_i : Y_i \to \mathbb{R}$ as $w_i(\mathbf{c}) := |\text{cluster}(\mathbf{c}, P_i)|$, for every $\mathbf{c} \in Y_i$.
4: Collect $\{Y_i\}_{i\in\mathcal{R}}$ from the non-straggling nodes.
5: Let $Y := \cup_{i\in\mathcal{R}} Y_i$. Using $\mathbf{b}$ from (1), define $w : Y \to \mathbb{R}$ such that[1] $w(\mathbf{c}) = b_i \cdot w_i(\mathbf{c})$ for all $\mathbf{c} \in Y_i$ and $i \in \mathcal{R}$.
6: Return $\widehat{C}$, the k-median cluster centers of Y.

Before assessing the quality of $\widehat{C}$ on the entire dataset P, we show that for any set of k centers C, the cost incurred by the weighted dataset Y is close to the cost incurred by P.

Lemma 2. *For any set of k-centers $C \subset \mathbb{R}^d$*

$$\text{cost}(P, C) - \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) \leq \text{cost}(Y, C, w) \leq 2(1+\delta)\text{cost}(P, C).$$

The following result quantifies the quality of the clustering solution returned by Algorithm 1 on the entire dataset P.

Theorem 3. *Let C^* be the optimal set of k-median centers for the dataset P. Then, $\text{cost}(P, \widehat{C}) \leq 3(1+\delta)\text{cost}(P, C^*)$.*

Proof. Using the lower bound from Lemma 2 with $C = \widehat{C}$,

$$\begin{aligned}\text{cost}(P, \widehat{C}) &\leq \text{cost}(Y, \widehat{C}, w) + \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) \\ &\overset{(i)}{\leq} \text{cost}(Y, C^*, w) + \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C^*) \\ &\overset{(ii)}{\leq} 2(1+\delta)\text{cost}(P, C^*) + (1+\delta)\text{cost}(P, C^*),\end{aligned}$$

where (i) follows from the fact that $\widehat{C}$ and Y_i are the optimal set of centers for the weighted dataset (Y, w) and the partial dataset P_i, respectively. For (ii), we utilize the upper bound in Lemma 2 and Lemma 1 (with $C = C^*$). □

In Algorithm 1, each compute node sends clustering solution on its local data using which the coordinator is able to construct a good summary of the entire dataset P despite the presence of the stragglers. This summary is sufficient to generate a good quality k-median clustering solution on P. In Section III-C, we show that if each compute node sends more information in the form of a coreset of its local data, the accumulated information at the coordinator is sufficient to solve the more general problem of (r, k)-subspace clustering in a straggler-resilient manner.

C. Straggler-resilient distributed (r, k)-subspace clustering

In this subsection, we utilize our redundant data assignment with straggler-resilient property to combine local coresets received from the non-straggling nodes to obtain a global coreset for the entire dataset, which further enables us to

[1]If for some $i_1 \neq i_2 \cdots \neq i_u$, $\mathbf{c} \in Y_{i_1} \cap Y_{i_2} \cdots \cap Y_{i_u}$, then we define $w(\mathbf{c}) = b_{i_1} \cdot w_{i_1}(\mathbf{c}) + b_{i_2} \cdot w_{i_2}(\mathbf{c}) \cdots + b_{i_u} \cdot w_{i_u}(\mathbf{c})$.

perform distributed (r,k)-subspace clustering in a straggler-resilient manner. In particular, we propose two approaches to perform subspace clustering, which rely on the coresets [12] and the *relaxed* coresets [11], [13], respectively.

1) Distributed (r,k)-subspace clustering using coresets: Here, we propose a distributed (r,k)-subspace clustering algorithm that used the existing coreset constructions from the literature in a black-box manner. Each compute node sends a coreset of its partial data which when re-weighted according to Lemma 3 gives us a coreset for the entire dataset even in presence of stragglers. Given this global coreset, we can then construct a solution to the underlying (r,k)-subspace clustering problem at the coordinator (cf. Theorem 1). The complete description of this approach is given in Algorithm 2.

Algorithm 2 Straggler-resilient (r,k)-subspace clustering

1: Input: A collection of n vectors $P \subset \mathbb{R}^d$.
2: Allocate P to s workers according to A with Property 1.
3: For each $i \in [s]$, find a δ-coreset (S_i, w_i) for P_i.
4: Collect $\{S_i\}_{i\in\mathcal{R}}$ at the coordinator.
5: Let $S = \cup_{i\in\mathcal{R}} S_i$. For every $i \in \mathcal{R}$, scale the weights of the coreset points received from i-th node by b_i (cf. (1)) i.e., $w(\mathbf{c}) = b_i w_i(\mathbf{c})\ \forall \mathbf{c} \in S_i$.
6: Return $\widehat{C}$, the set of r-subspaces that is an α-approximate solution to the (r,k)-subspace clustering on input (S,w).

Before we analyze the quality of $\widehat{C}$, the solution returned by Algorithm 2, we present the following result that shows the utility of an assignment scheme with Property 1 to construct a global coreset for the entire dataset from the coresets of the partial datasets in a straggler-resilient manner.

Lemma 3. *Let $P \subset \mathbb{R}^d$ be distributed according to A with Property 1. Let $\mathbf{b} \in \mathbb{R}^{|\mathcal{R}|}$ be the recovery vector for the set of non-straggler nodes $\mathcal{R} \subset [s]$. For any $i \in \mathcal{R}$, let S_i be an ϵ-coreset for the local dataset P_i with weight function $w_i : P_i \to \mathbb{R}$ with respect to the cost function $\mathrm{cost}(\cdot,\cdot)$. Then, $S := \cup_{i\in\mathcal{R}} S_i$ with the weight function $w : S \to \mathbb{R}$ defined as $w(\mathbf{c}) = w_i(\mathbf{c}) \cdot b_i$ for all $\mathbf{c} \in S_i$ is a $2(\epsilon+\delta)$-coreset for P.*

Since each S_i is a δ-coreset for S_i, it follows from Lemma 3 that S is a 4δ-coreset for P. This allows us to quantify the quality of $\widehat{C}$ as a solution to the underlying problem of (r,k)-subspace clustering on P.

Theorem 4. *Let* OPT *be the cost of the optimal (r,k)-subspace clustering solution for P. Then, we have that $\mathrm{cost}(P,\widehat{C}) \le \alpha(1+8\delta)\cdot \mathrm{OPT}$.*

Proof. Let C^* be the optimal (r,k)-subspace clustering solution for P, i.e., $\mathrm{cost}(P,C^*) = \mathrm{OPT}$. Since S is a 4δ-coreset, we have

$$\mathrm{cost}(P,C) \overset{(i)}{\le} \frac{\mathrm{cost}(S,C,w)}{(1-4\delta)} \overset{(ii)}{\le} \frac{\alpha}{(1-4\delta)}\mathrm{cost}(S,C^*,w) \overset{(iii)}{\le} \alpha\frac{(1+4\delta)}{(1-4\delta)}\mathrm{cost}(P,C^*) \le \alpha(1+8\delta)\mathrm{OPT},$$

where (i) and (iii) follow from Definition 2; and (ii) follows from the fact that the coordinator performs an α-approximate subspace clustering on (S,w). □

Coreset constructions for various clustering algorithms with squared ℓ_2 error was considered by [12], [14]–[21]. However, the size of such constructed coresets depends on the ambient dimension d which makes them prohibitive for high-dimensional datasets. Interestingly, Feldman et al. [11] and Balcan et al. [13] show that one could construct smaller sized relaxed coresets (of size independent of both n and d) for the r-PCA problem. Further, they use these relaxed coresets for approximate-PCA to solve the class of (r,k)-subspace clustering problems. Below we show that, by utilizing a redundant assignment scheme with Property 1, the distributed approximate PCA and hence, (r,k)-subspace clustering algorithms of [11], [13] can be made straggler resilient.

2) Straggler-resilient PCA using relaxed coresets: In this section, we use P to denote both the set of n data points and the $n \times d$ matrix with n points in P as its rows. We use the set and matrix notation interchangeably through this section.

The goal in PCA is to find the linear r-dimensional subspace L, that *best fits* the data. It is well-known that the subspace spanned by the top r right singular vectors of P gives an optimal solution to the PCA problem. The main question addressed by Feldman et al. [11] is to find an approximate solution to PCA in a distributed setting by constructing relaxed coresets for the local data. In Algorithm 3, we adapt the distributed PCA algorithm of [11] to obtain a straggler-resilient distributed PCA algorithm.

Algorithm 3 Straggler-resilient distributed r-PCA

1: Input: An $n \times d$ matrix P
2: Allocate P to s workers according A with Property 1.
3: Let P_i denote the sub-matrix of P contained at node i. Compute the SVD $P_i = U_i\Sigma_i V_i^T$.
4: For $r_1 = r + r/\delta - 1$, define $\Sigma_i^{(r_1)}$ to be the matrix that contains first r_1 diagonal entries of Σ_i, and 0 othewise.
5: Send $S_i = \Sigma_i^{(r_1)} V_i^T$.
6: Stack the S_i's received from non-stragglers to construct $Y = [S_i]_{i\in\mathcal{R}}$ and define $w(\mathbf{y}) = b_i$, for $\mathbf{y} \in S_i$.
7: Let $S = U\Sigma V^T$.
8: Return $\widehat{L} = r\text{-PCA}(S,w)$.

The following result from [11] shows that each S_i in Algorithm 3 is a relaxed δ-coreset of P_i. Note the additive Δ_i term which is not present in the original definition of a coreset (cf. Definition 2).

Lemma 4 ([11])**.** *For any $i \in [s]$, let $\widehat{P}_i$ denote the rows of $U_i\Sigma_i^{(r_1)}V_i^T$. Then, for any r-dimensional linear subspace $L \subset \mathbb{R}^d$, there exists $\Delta_i = \Delta_i(P_i,\widehat{P}_i) > 0$ such that*

$$\mathrm{cost}(P_i,L) \le \mathrm{cost}(\widehat{P}_i,L) + \Delta_i \le (1+\delta)\mathrm{cost}(P_i,L).$$

Note that, for each $i \in [s]$, only the first r_1 rows of S_i in Algorithm 3 are non-zero; as a result, each local relaxed coreset consists of r_1 vectors. Next, we show an equivalent of Lemma 1 for relaxed coresets.

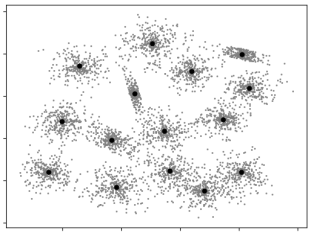

Fig. 1: Ground-truth.

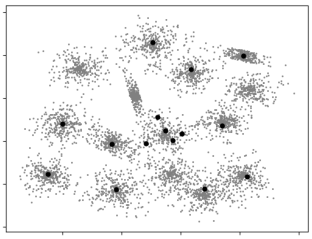

Fig. 2: No redundancy.

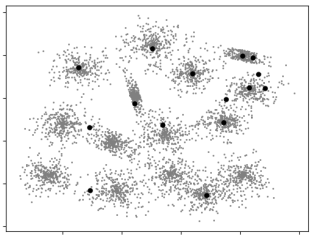

Fig. 3: $p_a = 0.1$.

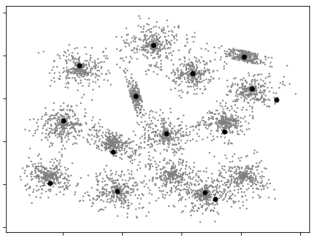

Fig. 4: $p_a = 0.2$.

Lemma 5. *Let $\Delta := \sum_{i \in \mathcal{R}} b_i \Delta_i$. Then, for any r-dimensional linear subspace $L \subset \mathbb{R}^d$,*

$$\text{cost}(P, L) \leq \text{cost}(Y, L, w) + \Delta \leq (1 + 4\delta)\text{cost}(P, L).$$

Now, Lemma 5 along with Lemma 4 enable us to guarantee that the solution obtained from Algorithm 3 is close to the optimal for the distributed PCA problem.

Theorem 5. *Let L^* be the optimal solution to r-PCA on P. Then,* $\text{cost}(P, \widehat{L}) \leq (1 + 4\delta)\text{cost}(P, L^*)$.

Proof. Note that Δ is independent of the choice of L. Thus,

$$\begin{aligned}\text{cost}(P, \widehat{L}) &\overset{(i)}{\leq} \text{cost}(Y, \widehat{L}, w) + \Delta \\ &\overset{(ii)}{\leq} \text{cost}(Y, L^*, w) + \Delta \overset{(i)}{\leq} (1 + 4\delta)\text{cost}(P, L^*),\end{aligned}$$

where (i) and (iii) follow from Lemma 5; and (ii) follows as $\widehat{L}$ is the optimal solution to the r-PCA problem on Y. □

D. Construction of assignment matrix

Finally, we present a randomized construction of the assignment matrix that satisfies Property 1. For the construction, we assume a random straggler model, where every compute node behaves as a straggler independently with probability p_t. Therefore, we receive the local computation from each compute node with probability $1 - p_t$.

Consider the following random ensemble of assignment matrices such that for some ℓ (to be chosen later) the (i, j)-th entry of the assignment matrix is defined as

$$A_{i,j} = \begin{cases} 1 & \text{with probability } p_a = \frac{\ell}{s}, \\ 0 & \text{with probability } 1 - p_a. \end{cases} \tag{2}$$

We show that for an appropriate choice of ℓ, and hence p_a, the random matrix A satisfies Property 1 with high probability.

Theorem 6. *For any $\delta > 0$, the randomized assignment matrix (cf. (2)) with $\ell = \frac{6(2+\delta)^2}{\delta^2} \cdot \frac{\log(\sqrt{2}n)}{1-p_t}$ satisfies Property 1 with probability at least $1 - \frac{1}{n}$ under the random straggler model.*

The two parameters of importance when constructing an assignment matrix are the *load per machine* and the *fraction of stragglers* that can be tolerated. Increasing the redundancy makes the assignment matrix robust to more stragglers while at the same time, increases the computational load on individual compute nodes. For $s = O(n)$, our construction assigns $O(\log n)$ data points to each compute node and is resilient to a constant fraction of random stragglers.

IV. Experiments

In this section, we demonstrate the performance of our straggler-resilient distributed k-medians algorithm and compare it with non-redundant data assignment scheme. We consider the synthetic Gaussian data-set [22] with $n = 5000$ two-dimensional points. The points are distributed on $s = 10$ compute nodes, $t = 3$ of which are randomly chosen to be stragglers. The results are presented in Figures 1-4.

Figures 1 shows the ground truth $k = 15$-median clustering, using the centroids provided in the data-set. Figures 2 shows the results obtained by ignoring the local computations from the straggler nodes. We use Algorithm 1 without any redundant data assignment. The 5000 data points are randomly partitioned among 10 compute nodes. Each non-straggler compute node sends its local k-median centers to the coordinator. The coordinator then runs a k-median algorithm on the accumulated $k(s - t)$ centers obtained from the non-stragglers. As evident from the comparison between Figure 1 and Figure 2, such a scheme can output a set of poor quality k-centers.

Figure 3 shows the result of Algorithm 1. The assignment matrix is chosen randomly (see Section III-D for details) with each $p_a := Pr[A_{i,j} = 1] = 0.1$. Such an assignment matrix assigns 500 data points to each compute node in expectation, leading to a non-redundant data assignment. Figure 4 shows the effect of increasing p_a to 0.2, and hence the redundancy in the data assignment step. Each compute node now gets about 1000 data points. Note that the results of Figure 4 are very close to the ground truth clustering shown in Figure 1.

V. Conclusion and Future directions

It is an interesting direction to explore the tradeoff between the communication and the approximation factor achieved by the clustering method. In the k-median algorithm described above, each compute node returns a set of k-centers to achieve an approximation factor of about 3. Whereas, in Algorithm 2, the compute nodes send the coresets of their local data which can then be combined to construct a global coreset for the entire data. While, the quality of the obtained centers improves, it increases the communication cost between the compute nodes and the coordinator since a coreset would contain more than k points.

Another natural question that we are currently exploring is using data distribution techniques to design distributed algorithms that are robust to byzantine adversaries.

REFERENCES

[1] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 359–366, 2000.

[2] P. Awasthi, M. F. Balcan, and C. White. General and robust communication-efficient algorithms for distributed clustering. *CoRR*, abs/1703.00830, 2017.

[3] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1063–1071, 2015.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, March 2018.

[5] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *Proceedings of the 34th International Conference on International Conference on Machine Learning (ICML)*, pages 3368–3376, 2017.

[6] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd. In *Proc. of the AISTATS*, pages 803–812, 2018.

[7] Zachary Charles, Dimitris Papailiopoulos, and Jordan Ellenberg. Approximate gradient coding via sparse random graphs. *arXiv preprint arXiv:1711.06771*, 2017.

[8] C. Karakus, Y. Sun, S. Diggavi, and W. Yin. Straggler mitigation in distributed optimization through data encoding. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*, pages 5440–5448, 2017.

[9] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr. Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding. In *Proceedings of 2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2022–2026, June 2018.

[10] S. Dutta, V. Cadambe, and P. Grover. Short-dot: Computing large linear transforms distributedly using coded short dot products. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*, pages 2100–2108, 2016.

[11] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1434–1453, 2013.

[12] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*, pages 291–300. ACM, 2004.

[13] Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, pages 3113–3121, 2014.

[14] Sariel Har-Peled. No, coreset, no cry. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 324–335. Springer, 2004.

[15] Michael Edwards and Kasturi Varadarajan. No coreset, no cry: Ii. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 107–115. Springer, 2005.

[16] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of Computing*, pages 209–217. ACM, 2005.

[17] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

[18] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

[19] Michael Langberg and Leonard J Schulman. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM Symposium on Discrete Algorithms*, pages 598–607. SIAM, 2010.

[20] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM Symposium on Theory of Computing*, pages 569–578. ACM, 2011.

[21] Kasturi Varadarajan and Xin Xiao. A near-linear algorithm for projective clustering integer points. In *Proceedings of the twenty-third annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1329–1342. SIAM, 2012.

[22] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006.