Abstract

Distributed clustering has attracted significant attention in recent years. In this paper, we study the $k$-means problem in the distributed dimension setting, where the dimensions of the data are partitioned across multiple machines. We provide new approximation algorithms, which incur low communication costs and achieve constant approximation ratios. The communication complexity of our algorithms significantly improve on existing algorithms. We also provide the first communication lower bound, which nearly matches our upper bound in a certain range of parameter setting. Our experimental results show that our algorithms outperform existing algorithms on real data-sets in the distributed dimension setting.

1. Introduction

Clustering is a central problem in unsupervised learning and data analytics. Among hundreds of clustering problems and algorithms, the $k$-means problem is arguably the most popular one, due to its simplicity and effectiveness in numerous applications. Specifically, given $n$ points in $\mathbb{R}^d$, $k$-means seeks $k$ points (cluster centers) to induce a partition of the given point-set, so that the average squared distance from each data point to its closest cluster center is minimized.

Due to the sheer increase in volume and dimensionality of data, clustering (in particular $k$-means) in the distributed setting has attracted significant attention from various research communities in recent years. While much prior work has focused on the model where the data records (points, items) are distributed in different locations, we study the $k$-means problem in the distributed dimension model, in which the dimensions of the data points are stored across multiple machines.

Suppose the input data points are vectors in $\mathbb{R}^d$. In the distributed dimension model, each party is assigned a subset of the $d$ dimensions, and holds the projection of all the data points onto these dimensions. If the whole input point-set is represented by an $n \times d$ matrix, where each row indicates one point and each column indicates one dimension, each party holds a column partition in our model, while the prior work focused on row partition. The distributed dimension (or column partition) model arise naturally in a variety of applications and data management platforms, some of which we briefly mention below.

(1). Suppose that we want to cluster online users based on several features such as online shopping records, online social relationships, and job hunting records, which are collected by different companies (Amazon, Facebook, and Linkedin) (assuming that there is a unique id, e.g., email address, which can be used to identify the same user). It is also quite common that different teams in the same company develop apps which automatically store different features in different machines. (2). Several widely deployed distributed DBMS systems, such as MonetDB, C-Store, VectorWise, HBASE, are column-oriented by design (Stonebraker et al., 2005; Abadi et al., 2009; 2013). These systems partition and store the data based on the columns. (3). Another motivation comes from the proliferation of networked sensing systems, where we need to monitor several attributes of the objects using multiple sensors of different types. For example, an environment monitoring system (Gu et al., 2005; Su et al., 2011) may have video cameras, and different sensors recording audio, temperature, il-
1. Formulation

We formally introduce some notations and our communication model. We use $[n]$ as a short-hand notation for $\{1, 2, \ldots, n\}$. Let $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^d$ be the set of points we want to cluster. There are $T$ parties. The $l$-th party holds a subset $D_l$ of dimensions. Assume that $\bigcup_{l=1}^{T} D_l = [d]$ and $\sum_{l=1}^{T} d_l = d$ (i.e., $\{D_l\}_{l=1}^{T}$ is a partition of $[d]$). For each point $p_i \in \mathbb{R}^d$, we can write $p_i = (p^1_i, p^2_i, \ldots, p^T_i)$, where $p^l_i \in \mathbb{R}^{d_l}$ ($1 \leq l \leq T$). Let $P_l = \{p^l_i\}_{i \in [n]}$ be the data stored in the $l$-th party. With a slight abuse of notation, we also use $P$ and $P_l$ to denote their induced $n \times d$ and $n \times d_l$ matrices. Given a small integer $k \geq 1$, our goal is to find $k$ cluster centers and a partition of $P$ in the central server, such that the corresponding $k$-means clustering cost is minimized, and the communication complexity is as small as possible.

Communication model: We assume that there is a central server (or server in short) and only allow the communication between the central server and each of the $T$ parties. We only require the central server to know the final solution. See Fig. 1 for an illustration. For simplicity, the communication cost are measured in words, i.e., sending one single number costs 1. The communication complexity/cost of an algorithm is the total number of words transmitted between the server and the parties. We also briefly discuss the communication complexity measured in bits in Section 7.

To capture the high volume and dimensionality of the data, we should think that $n$ and $d$ are very large, while $k$ and $T$ are much smaller in typical applications and treated as constants.

Note: To measure the quality of the clustering result, we compute the ratio between the obtained and minimum clustering costs, which is called approximation ratio (the closer to 1, the better).

1.2. Prior Work

We are not aware of any prior work explicitly focusing on clustering in distributed dimension setting (most existing algorithms for distributed clustering are for the row partition model, or some other purposes such as privacy preserving or making a consensus from different subspaces; the reader is referred to the end of this subsection for the details of these results), however, some can be easily adopted, which we briefly sketch below.

Random projection is a powerful technique for handling high dimensional data, and has been studied by (Boutsidis et al., 2015; Cohen et al., 2015) in the distributed clustering context. We can use random projection in the distributed dimension setting as follows: We multiply the sub-matrix $P_l$ by a sub-matrix of the unified random projection matrix in the $l$-th party (see Fig. 2). Once the central server obtains $P \times R$ by receiving $P_l \times R_l$ from each party, it can perform any existing $k$-means clustering algorithm on it. The random projection ensures that the cost of $k$-means clustering is approximately preserved so that the performance guarantee of the distributed algorithm is almost the same as the centralized one.

[Diagram of communication model]

SVD can also be applied to $k$-means with distributed dimensions. There are two different ways to use SVD. (1) In the first approach, the server computes the orthonormal basis of a low dimensional subspace for approximating $P$ (Feldman et al., 2013; Liang et al., 2014; Boutsidis et al., 2015; Cohen et al., 2015) (i.e., approximate PCA) by computing the SVD of $P$ in the distributed setting$^2$, and

$^2$Existing distributed SVD algorithms work for both the row partition and column partition models. We exchange the roles of $n$ and $d$ here due to column partition. So the basis now is a $n \times t$ matrix where $t$ depends on each individual method.
Other Related Work. A number of results for clustering under centralized setting have been obtained in the past (Aggarwal & Reddy, 2013; Awasthi & Balcan, 2014). Particularly, k-means++ (Arthur & Vassilvitskii, 2007) and some of its variants are very efficient in practice and achieve the expected approximation ratio \(O(\log k)\). Recently, (Chawla & Gionis, 2013) develops an algorithm for k-means with outliers. (Feldman et al., 2012; Zhang et al., 2008) apply coresets (Feldman & Langberg, 2011) to handle the problem of clustering with distributed data points (row partition). Following their works, (Balcan et al., 2013) develops the algorithm applicable to any connecting topology of communication networks, and also improves the communication cost. Several other algorithms for distributed data points are also proposed (Datta et al., 2006; Forman & Zhang, 2000; Bahmani et al., 2012). Some clustering results for distributed dimensions (Vaidya & Clifton, 2003; Jagannathan & Wright, 2005) mainly focus on privacy preserving rather than communication cost. Another similar problem is called multi-view clustering (Bickel & Scheffer, 2004) which focuses more on how to achieve a consensus of the multiple views, and the computation is usually executed in a single machine without concerning communication cost. Besides distributed clustering, many other machine learning problems are recently studied under distributed setting, such as PCA (Liang et al., 2014; Kannan et al., 2014), classification (Daumé III et al., 2012), consensus of the multiple views, and the computation is usually executed in a single machine without concerning communication cost. Besides distributed clustering, many other machine learning problems are recently studied under distributed setting, such as PCA (Liang et al., 2014; Kannan et al., 2014), classification (Daumé III et al., 2012), PAC (Balcan et al., 2012), and sparse learning (Bellet et al., 2015).

1.3. Our Contributions

In this paper, we provide communication-efficient algorithms for k-means in the distributed dimension setting, improving upon the previous methods. In the random projection method, each party multiplies a sub-matrix \(R_i\) to its own sub-matrix \(P_i\) (as Fig. 2). \(R_i\) is chosen in a data oblivious way, without utilizing the property of each \(P_i\). However, our method examines the distribution of \(P_i\) in each party, which allows us to spend less communication in our setting. In a high level, our algorithm works as follows. First, the \(l\)-th party computes a k-means clustering for \(P_i\) in the corresponding subspace \(\mathbb{R}^d\), and sends to the server the obtained k cluster centers, as well as the clustering memberships of the n points. Based on such information, the central server can approximate the point-set \(P\) by constructing a weighted grid in the whole space \(\mathbb{R}^d\). Finally, one can apply any existing k-means clustering algorithm to the grid. When \(k = O(1)\), we can achieve an approximation ratio of \(9 + \epsilon\), for any constant \(\epsilon > 0\), match-

<table>
<thead>
<tr>
<th>Reference</th>
<th>Approximation ratio</th>
<th>Communication complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson-Lindenstrauss (JL) transform*</td>
<td>(1 + \epsilon)</td>
<td>(O(kd))</td>
</tr>
<tr>
<td>(Boutsidis et al., 2015)†</td>
<td>(2 + \epsilon)</td>
<td>(O((2k + 2d) + Tn))</td>
</tr>
<tr>
<td>(Boutsidis et al., 2015)‡</td>
<td>(3 + \epsilon)</td>
<td>(O(kd))</td>
</tr>
<tr>
<td>(Boutsidis et al., 2015)*</td>
<td>(2 + \epsilon)</td>
<td>(O(kd))</td>
</tr>
<tr>
<td>(Liang et al., 2014; Cohen et al., 2015)†</td>
<td>(1 + \epsilon)</td>
<td>(O((2k + 2d) + Tn))</td>
</tr>
<tr>
<td>(Cohen et al., 2015)‡</td>
<td>(1 + \epsilon)</td>
<td>(O(kd))</td>
</tr>
<tr>
<td>(Cohen et al., 2015)*</td>
<td>(9 + \epsilon)</td>
<td>(O(kd))</td>
</tr>
<tr>
<td>Our results I</td>
<td>(9 + \epsilon)</td>
<td>(Tn + kd)</td>
</tr>
<tr>
<td>Our results II</td>
<td>(5 + \epsilon)</td>
<td>(2Tn + kTn + kd)</td>
</tr>
</tbody>
</table>

Table 1. Existing and our new results for k-means clustering with distributed dimensions (*: random projection, †: PCA, ‡: feature selection), where \(\epsilon > 0\). The communication complexities for random projection and feature selection all contain an additive term \(kd\). In fact, this is because once the central server obtains the clustering result in the space with reduced dimensionality, in order to map the k cluster centers back to the original space \(\mathbb{R}^d\), each party should send the coordinate of each cluster center in its reserved subspace to the server, incurring an extra communication cost of \(\sum_{i=1}^{Tn} kd_i = kd\).
In this section, we present a simple algorithm yielding a $\epsilon$-approximation ratio with a slightly larger communication complexity (Section 3). We show the comparison between the existing and our results for $k$-means clustering with distributed dimensions in Table 1. Note that the communication complexities from random projection and feature selection all have a hidden constant (usually bigger than 1) which depends on the success probability of the algorithms (due to their randomness). On the other hand, our communication protocol is deterministic, except that the centralized $k$-means algorithm (which is used as a black-box in each individual machine) may be randomized.

Furthermore, we point out another advantage of our method here. We can readily extend the method to some other related problems, such as $k$-means with outliers (Chawla & Gionis, 2013) and constrained $k$-means clustering (which has the same objective function as $k$-means but has extra combinatorial constraint) (Wagstaff et al., 2001; Ding & Xu, 2015; Bhattacharya et al., 2016), in the distributed dimension setting. We can achieve similar approximation ratios and communication complexities. The basic idea is that the built grid can be easily modified to adapt these problems, e.g., slightly increasing the size of the grid or sending more information besides the clustering memberships to the server. However, it is not clear to us how to adopt the existing approaches mentioned in Section 1.2 to these problems.

2. $(9 + \epsilon)$-Approximation

In this section, we present a simple algorithm yielding a $(9 + \epsilon)$-approximation for $k$-means clustering with distributed dimensions. The key idea is building a weighted grid in the original space $\mathbb{R}^d$ to approximately reconstruct the point-set $P$. The reason for using a grid is that it can incur significant saving in communication complexity under the problem setting, i.e., each party only needs to send a small amount of information to the central server for building such a grid. Moreover, in order to have a more accurate reconstruction, we assign a weight to each individual grid point. See Algorithm 1 for the details. In the algorithm, we denote $[k] \times [k] \times \cdots \times [k]$ as $[k]^T$. We also denote the distributed algorithm for solving the $k$-means problem by $A$, and assume that $A$ can achieve an approximation ratio of $\lambda (> 1)$. For $k = O(1)$, it is known that there is a PTAS (i.e., polynomial-time approximation scheme) for $k$-means (Kumar et al., 2010; Feldman & Langberg, 2011) which indicates that the approximation ratio $\lambda$ can be taken to be $1 + \epsilon$ for any constant $\epsilon > 0$. A toy example of the grid construction is shown in Fig. 3, and the theoretical guarantee is presented in the following Theorem 1.

\[ g(1,1,2) = (s_1, s_2^1, s_2^2). \]

**Theorem 1.** If we have a centralized algorithm $A$ for $k$-means clustering which can achieve an approximation ratio $\lambda$, DistDim-K-Means (Algorithm 1) yields a $(2\lambda^2 + 3\lambda + 2\lambda\sqrt{2(\lambda + 1)})$-approximation with communication cost $Tn + kd$. In particular, the approximation ratio is $9 + \epsilon$ if $A$ is a PTAS.

**Proof.** In DistDim-K-Means, only step 2 needs communication, and each party costs $n + kd_l$. Thus, the total communication cost is $Tn + k \sum d_l = Tn + kd$.

**Figure 3.** Let $n = 6$, $k = 2$, and $T = 3$. Suppose the clustering results $M_l$ for $l = 1, 2, 3$ in the three parties are $\{{p_1, p_2, p_3}, \{p_4, p_5, p_6}\}$, $\{{p_1, p_3}, \{p_2, p_4, p_5, p_6\}\}$, and $\{{p_1, p_2, p_4, p_5}, \{p_3, p_6\}\}$ respectively. Then the grid built by Algorithm Grid is shown in the figure. For example, the grid point $g(1,1,2) = (s_1, s_2^1, s_2^2)$. And since $M_1 \cap M_2 \cap M_3 = \{p_3\}$, the weight of $g(1,1,2)$ is 1.

**Figure 4.** Illustration for the utilizations of triangle inequality in (1) and (4). Below, we focus on the proof of the approximation guarantee. For simplicity, the grid $G$ is rewritten as $\{g_1, \ldots, g_m\}$.
Algorithm 1 DISTDIM-K-MEANS

1. For each $l \in [T]$, the $l$-th party does the following: runs the centralized approximation algorithm for k-means on $P_l$, and obtains the clustering memberships $M^l = \{M^l_1, \ldots, M^l_k\}$ on $[n]$ (i.e., each $M^l_i \subseteq [n]$ and $M^l_i \cap M^l_i' = \emptyset$ if $i \neq i'$) and a set of corresponding cluster centers $\{s^l_1, \ldots, s^l_k\} \subseteq \mathbb{R}^d$.

2. For each $l \in [T]$, the $l$-th party sends the memberships $M^l$ and the $k$ cluster centers to the server.

3. The server builds a weighted grid $G$ in $\mathbb{R}^d$ of size $k^T$, using Algorithm 2.

4. The server runs algorithm $A$ on $G$, and obtains the $k$ cluster centers $\{c^G_1, \ldots, c^G_k\}$ and the clustering memberships $M^G = \{M^G_1, \ldots, M^G_k\}$ on $[k]^T$.

5. The server obtains the final clustering memberships $M = \{M_1, \ldots, M_k\}$ on $[n]$, where each $M_j = \bigcup\{i \in \bigcap_{l=1}^T M^l_{i_l} \mid (i_1, \ldots, i_T) \in \mathcal{M}^G_j\}$.

6. Now, the server has the $k$ cluster centers $\{c^G_1, c^G_2, \ldots, c^G_k\}$ and the clustering memberships $M = \{M_1, \ldots, M_k\}$.

Algorithm 2 GRID

1. For each index tuple $(i_1, \ldots, i_T) \in [k]^T$, construct a point $g(i_1, \ldots, i_T) = (s^1_{i_1}, s^2_{i_2}, \ldots, s^T_{i_T}) \in \mathbb{R}^d$.

2. Output $G = \{g(i_1, \ldots, i_T) \mid (i_1, \ldots, i_T) \in [k]^T\}$, and each $g(i_1, \ldots, i_T)$ is associated with the weight $|\bigcap_{l=1}^T M^l_{i_l}|.$

where $m = k^T$, and for each $g_j$, its corresponding intersection $\bigcap_{l=1}^T M^l_{i_l}$ is rewritten as $S_j$. Suppose the $k$ cluster centers in an optimal solution are $\{c^*_1, \ldots, c^*_k\}$, and for each $p_i \in P$, we denote its nearest neighborhood from $\{c^*_1, \ldots, c^*_k\}$ as $N^G(p_i)$. Similarly, we denote the nearest neighborhood of each $g_j \in G$ from $\{c^G_1, \ldots, c^G_k\}$ as $N^G(g_j)$. Then, for each index $i \in S_j$,

$$\left\|p_i - N^G(g_j)\right\|^2 = \left\|p_i - g_j + g_j - N^G(g_j)\right\|^2 \leq \left\|p_i - g_j\right\|^2 + \left\|g_j - N^G(g_j)\right\|^2 + 2\left\|p_i - g_j\right\| \cdot \left\|g_j - N^G(g_j)\right\|. \quad (1)$$

See Fig. 4. For ease of analysis, we denote the cost of our solution from Algorithm 1, $\sum_{j=1}^k \sum_{i \in S_j} \left\|p_i - N^G(g_j)\right\|^2$, by $\Gamma$, and the cost of the optimal solution, $\sum_{j=1}^k \left\|p_i - N^G(g_j)\right\|^2$, by $\Gamma_{opt}$, respectively. Furthermore, $\sum_{j=1}^k \sum_{i \in S_j} \left\|p_i - g_j\right\|^2$ and $\sum_{j=1}^k |S_j| \left\|g_j - N^G(g_j)\right\|^2$ are denoted by $\Gamma_a$ and $\Gamma_b$, respectively. Summing both sides of (1), we have

$$\Gamma = \sum_{j=1}^k \sum_{i \in S_j} \left\|p_i - N^G(g_j)\right\|^2 \leq \Gamma_a + \Gamma_b + 2 \sum_{j=1}^k \sum_{i \in S_j} \left\|p_i - g_j\right\| \cdot \left\|g_j - N^G(g_j)\right\| \leq \Gamma_a + \Gamma_b + 2 \sqrt{\Gamma_a \Gamma_b}, \quad (2)$$

where the last inequality follows from Cauchy-Schwarz inequality. Note that $\{c^G_1, \ldots, c^G_k\}$ is a $\lambda$-approximation for the weighted point-set $G$, and each $g_j$ has weight $|S_j|$. Actually each $g_j$ can be viewed as $|S_j|$ unweighted points colocated at the same place. Then $G$ can be viewed as a set of $\sum_j |S_j| = n$ unweighted points. Thus, we have

$$\Gamma_b = \sum_{j=1}^k |S_j| \left\|g_j - N^G(g_j)\right\|^2 \leq \lambda \sum_{j=1}^k \sum_{i \in S_j} \left\|g_j - N^G(p_i)\right\|^2, \quad (3)$$

by the relaxed triangle inequality (see Fig. 4). (3) and (4) together imply that

$$\Gamma_b \leq 2\lambda(\Gamma_a + \Gamma_{opt}). \quad (5)$$

Combining (2) and (5), we have

$$\Gamma \leq (2\lambda + 1)\Gamma_a + 2\lambda \Gamma_{opt} + 2 \sqrt{2\lambda \Gamma_a (\Gamma_a + \Gamma_{opt})}. \quad (6)$$

Meanwhile, since the $T$ subspaces among different parties are mutually orthogonal and disjoint (except the origin), the cost $\Gamma_{opt}$ is the sum of the costs of their projections in the $T$ subspaces. Also, in each subspace we run algorithm $A$ which can achieve a factor of $\lambda$. Consequently, we have that

$$\Gamma_a = \sum_{j=1}^k \sum_{i \in S_j} \left\|p_i - g_j\right\|^2 \leq \lambda \sum_{p_i} \left\|p_i - N^G(p_i)\right\|^2 = \lambda \Gamma_{opt}. \quad (7)$$

Plugging (7) into (6), we have

$$\Gamma \leq (2\lambda^2 + 3\lambda + 2\lambda \sqrt{2(\lambda + 1)})\Gamma_{opt}. \quad (8)$$

Since there is a PTAS for $k$-means when $k = O(1)$, we can take $\lambda = 1 + O(c^2)$, and obtain the approximation ratio of $9 + \epsilon$ for our problem. □
3. \((5 + \epsilon)\)-Approximation

Here, we follow the idea of \textsc{DistDim-K-Means} but modify the construction of \(G\) in \textsc{Grid} (Algorithm 2), to improve the approximation ratio from \(9 + \epsilon\) to \(5 + \epsilon\). We also show how to reduce the following increased communication complexity via the idea of random projection by (Cohen et al., 2015) in Section 3.1.

\textbf{\textsc{DistDim-K-Means-Improved}:} In step 2 of \textsc{DistDim-K-Means}, we do not need to send the coordinates of the \(k\) centers from each party to the server; instead, in the \textsc{Grid} algorithm, we update each \(g_{(i_1,\ldots,i_T)}\) by the mean point of \(\{p_i \mid i \in \bigcap_{l=1}^{T} M_{i_l}\}\). To achieve that, the server broadcasts the index-set \(S_j = \bigcap_{l=1}^{T} M_{i_l}\) to each party, and then each party sends \(\frac{1}{|S_j|} \sum_{p_i \in S_j} p_i\), i.e., the projection of the mean point on its subspace back to the server. Finally, through simple concatenating the server can obtain \(\frac{1}{|S_j|} \sum_{p_i \in S_j} p_i\), i.e., the mean point.

We consider the extra communication cost caused by this modification. Since each index from \([n]\) belongs one and only one of the \(k^T\) index-sets, broadcasting all these index-sets to the \(T\) parties cost a total communication complexity of \(Tn\). Secondly, it costs \(d\) for constructing each individual mean point of \(\{p_i \mid i \in \bigcap_{l=1}^{T} M_{i_l}\}\), and thus \(kd\) for all the \(k^T\) mean points. Note that our method only benefits the case of \(n > k^T\), otherwise, we just simply send all the data to the server which costs a communication complexity of \(nd\). Overall, the modification needs an extra communication cost of \(Tn + kd\).

**Theorem 2.** If we have a centralized algorithm \(A\) for \(k\)-means clustering which can achieve an approximation ratio \(\lambda\), \textsc{DistDim-K-Means-Improved} yields a \((2\lambda^2 + 3\lambda)\)-approximation with communication cost \(2Tn + k^Td\). The approximation ratio can be \(5 + \epsilon\) if \(A\) is a PTAS.

**Proof.** From the above analysis, we can easily obtain the communication complexity of \(2Tn + k^Td\). For the approximation ratio, we recall the proof of Theorem 1. At first, we need the well known fact below.

**Fact 1.** Let \(q^*\) be the mean of a point-set \(Q \subset \mathbb{R}^d\), then for any arbitrary point \(q \in \mathbb{R}^d\), we have
\[
\sum_{q_i \in Q} \|q - q_i\|^2 = \sum_{q_i \in Q} \|q^* - q_i\|^2 + |Q|\|q - q^*\|^2. \tag{9}
\]

Based on the above (9) and the fact that \(g_j\) is the mean of \(\{p_i \mid i \in S_j\}\), we rewrite the cost of our solution
\[
\Gamma = \sum_{j=1}^{k} \sum_{i \in S_j} \|p_i - N^G(g_j)\|^2
\]
\[
= \sum_{j=1}^{k} \sum_{i \in S_j} \|p_i - g_j\|^2 + |S_j|\|g_j - N^G(g_j)\|^2
\]
\[
= \Gamma_a + \Gamma_b \tag{10}
\]

Also, with respect to the value of \(\Gamma_a\), since we change the position of each \(g_j\) to the mean point of \(\{p_i \mid j \in S_j\}\), the value of \(\Gamma_a\) cannot increase. Furthermore, following the same manner of (3)-(7), we have that \(\Gamma \leq (2\lambda^2 + 3\lambda)\Gamma_{opt}\). Finally, the approximation ratio can reach \(5 + \epsilon\) if \(\lambda = 1 + O(\epsilon)\).

3.1. Improvement via Random Projection

(Cohen et al., 2015) shows that given an instance of \(k\)-means clustering, one can project all the points to a randomly selected \(O(k/\epsilon^2)\)-dimensional space, such that when recovering the solution in the original space, the approximation ratio increases by a factor at most \((1 + \epsilon)\). In this section, we show how to apply this result to improve the communication complexity of the \((5 + \epsilon)\)-approximation.

One expensive step of \textsc{DistDim-K-Means-Improved} is computing the exact coordinates of the grid points \(\{g_1, \ldots, g_{k^T}\} \in \mathbb{R}^d\), which costs a communication complexity of \(k^Td\). From (Cohen et al., 2015), we know that it is not necessary to do that. Actually, it is sufficient to only know their coordinates in \(\mathbb{R}^{O(k/\epsilon^2)}\) after random projection. To realize that, we can use exactly the same idea showed in Fig. 2. As a consequence, each party sends a \(k^T \times O(k/\epsilon^2)\) matrix to the server, and the communication complexity of this step is reduced from \(k^T d\) to \(O(T k^{2+1}/\epsilon^2)\) (suppose \(d \gg T k/\epsilon^2\)). In addition, after finding the clustering memberships of \(P\) in \(\mathbb{R}^{O(k/\epsilon^2)}\), we still need to compute the coordinates of the \(k\) cluster centers in \(\mathbb{R}^d\), which costs an extra \(Tn + kd\) in communication complexity.

For the final approximation ratio, we can slightly reduce the value of \(\epsilon\) used in the random projection and PTAS of centralized \(k\)-means clustering by a constant factor, such that the ratio of \(5 + \epsilon\) in Theorem 2 can still be guaranteed.

**Theorem 3.** Through random projection, the communication complexity of \textsc{DistDim-K-Means-Improved} is reduced to \(3Tn + kd + O(T k^{2+1}/\epsilon^2)\) and the same approximation ratio is preserved.

4. Other Results

We extend our \textsc{DistDim-K-Means} algorithm to some other related clustering problems with distributed dimensions. In general, we consider two problems motivated by the particular real-world applications, \(k\)-means clustering with outliers and constrained \(k\)-means clustering. To the best of our knowledge, there is no obvious way to extend the previous mentioned approaches, such as random projection or SVD, to these problems.

4.1. \(k\)-Means Clustering with Outliers

Comparing to the ordinary \(k\)-means clustering, its variant of that considering outliers is more general and robust in practice. Using the same notations defined before and an
additional integer parameter $z < n$, the object is to kick out $z$ points from $P$ such that the objective value of $k$-means clustering for the remaining points is as small as possible. For this problem, we still follow the idea and main structure of DistDim- $K$-Means. Below is our modification.

**DistDim-$K$-Means-Outliers.** We only need to modify a few places in DistDim-$K$-Means. At first, similar to $k$-means clustering, we assume that $B$ is the centralized algorithm solving $k$-means with outliers and achieving an approximation of $\lambda > 1$. In step 1, we run $B$. Moreover, the corresponding memberships $\mathcal{M}^t$ become $\{\mathcal{M}^t_1, \cdots, \mathcal{M}^t_{k+z}\}$ where the last $z$ items indicates the indices of the $z$ outliers. Also, in step 2 each party sends the corresponding memberships and coordinates of not only the $k$ centers but also the $z$ outliers. Consequently, the weighted grid $G$ has a size of $(k + z)^T$. Similarly, we run $B$ and compute the memberships in step 4 & 5. Note that when running $B$ in step 4, we view each $g_{(\ell_1, \cdots, \ell_T)}$ as a multi-set of $\bigcap_{\ell=1}^T \mathcal{M}^t_{i\ell}$ individual unweighted points. Finally, in step 5 we obtain the clustering memberships $\mathcal{M}$ on the $n$ points except $z$ outliers.

The theoretical guarantee is presented in the following Theorem 4. The proof is similar to that of Theorem 1, and due to space limit we put it in our supplement.

**Theorem 4.** If we have a centralized algorithm $B$ for $k$-means clustering with outliers which can achieve an approximation ratio $\lambda$, DistDim-$K$-Means-Outliers yields a $(2\lambda^2 + 3\lambda + 2\sqrt{2(\lambda + 1)})$-approximation with communication cost $Tn + (k + z)d$. In particular, the approximation ratio is $9 + \epsilon$ if $B$ is a PTAS.

**Remark 1.** Different from $k$-means, the only known PTAS for $k$-means with outliers (for the centralized setting) (Feldman & Schulman, 2012) requires the number of outliers to be a constant. For those applications where we need to exclude more outliers, we can instead run some heuristic algorithms such as (Chawla & Gionis, 2013).

### 4.2. Constrained $k$-Means Clustering

Given an instance of $k$-means clustering, many real-world applications do not allow the set of points to be arbitrarily assigned to the clusters. The constraints, such as upper or lower bound on the size of each cluster and chromatic conflict (e.g., the points sharing the same color cannot be clustered into the same cluster), are motivated from data mining (Wagstaff et al., 2001), resource assignment (Khuller & Sussmann, 1996), and privacy preserving (Sweeney, 2002; Aggarwal et al., 2010; Li et al., 2010). Recently, (Ding & Xu, 2015; Bhattacharya et al., 2016) provide PTAS for a large class of constrained $k$-means clustering when $k = O(1)$.

Actually, some of the constrained $k$-means clustering problems under the setting of distributed dimensions can be easily solved by the framework of our DistDim-$K$-Means in Section 2 (just replace the algorithm $A$ by the corresponding algorithms handling the constraints, such as the PTAS by (Ding & Xu, 2015; Bhattacharya et al., 2016), in the server and each party). To shed some light on this, consider $k$-means clustering with a lower bound $r > 0$ on the size of each cluster. Once building the grid $G$, we can still bound the distortion between $G$ and the original point-set $P$ by the similar way of (7), because the cost of the objective function (even after adding lower bound constraint) is always equal to the sum of the costs in the mutually orthogonal and disjoint subspaces due to the nature of $k$-means. Then through the manner of (2)-(6), we can obtain the same approximation ratio and communication complexity. Moreover, for item-level constraints such as chromatic conflict, we can view each grid point of $G$ as a set of individual colored points locating at the same location, and consequently $G$ is in fact a new instance of $k$-means clustering with chromatic conflict but having a bounded distortion to the original instance $P$; finally the theoretical results can be similarly guaranteed as well.

### 5. Lower Bound

In this section, we provide a lower bound of the communication cost for $k$-means problem with distributed dimensions. In fact, the lower bound even holds for the special case where the $t$-th party holds the $l$-th column. We denote by $k$-Means$_{n,T}$ the problem where there are $T$ parties and $n$ points in $\mathbb{R}^T$, and we want to compute $k$-means in the server. We prove a lower bound of $\Omega(n \cdot T)$ for $k$-Means$_{n,T}$ (for achieving any finite approximation ratio) by a reduction from the set disjointness problem (Chattopadhyay & Pitassi, 2010). Due to space limit, we put the details in our supplement.

### 6. Experiments

We compare our algorithms of DistDim-$K$-Means and DistDim-$K$-Means-Outliers to the following three types of methods as mentioned in Section 1.2: (1) the random projection based approach RP-$t$ where $t$ is the dimensionality of the random subspace, and the two SVD based approaches (2) PCA-$t$ where $t$ is the rank of the reconstructed matrix approximating $P$ and (3) FS-$t$ where $t$ is the number of selected features (dimensions).

**Data-sets & Experimental methodology.** We first choose a real-world data-set from (Bache & Lichman, 2013), YearPredictionMSD which contains $10^5$ points in $\mathbb{R}^{90}$. We randomly divide the data-set into 3 parties with each having 30 attributes (i.e., $T = 3$), and set $k = 500$-100. Also, for $k$-means clustering with outliers we set the number of outliers $z = 500$. In the server and each party, we use the algorithms from (Arthur & Vassilvitskii, 2007) and (Chawla & Gionis, 2013) as the centralized algorithms $A$ and $B$, i.e., the subroutines in DistDim-$K$-Means and DistDim-$K$-
K-Means Clustering with Distributed Dimensions

Figure 5. The normalized costs of k-means clustering on (a) YearPredictionMSD, (b) YearPredictionMSD considering outliers, and (c) Bag of Words(NYTimes).

<table>
<thead>
<tr>
<th></th>
<th>Our Alg</th>
<th>RP</th>
<th>PCA</th>
<th>FS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KMeans</td>
<td>6.02</td>
<td>8.87</td>
<td>32.30</td>
</tr>
<tr>
<td></td>
<td>Outlier</td>
<td>7.32</td>
<td>9.76</td>
<td>33.19</td>
</tr>
</tbody>
</table>

Table 2. Average communication costs for YearPredictionMSD data-set (in100KB) over the values of k.

<table>
<thead>
<tr>
<th></th>
<th>Our Alg</th>
<th>RP</th>
<th>PCA</th>
<th>FS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KMeans</td>
<td>20</td>
<td>27.54</td>
<td>33.30</td>
</tr>
<tr>
<td></td>
<td>Outlier</td>
<td>40</td>
<td>35.55</td>
<td>37.30</td>
</tr>
</tbody>
</table>

Table 3. Communication costs for NYTimes data-set (in MB).

Results & Analysis. Firstly, comparing to the clustering costs obtained by the centralized algorithms A and B (directly run on the whole data-sets without considering the distributed setting) for different values of $k$, our results obtained in the distributed setting are over within a factor of no more than 1.5 for most of the cases, which is much better than the theoretical ratio $9 + \epsilon$ by Theorem 1&4.

Secondly, we show the comparison on clustering cost between our and other algorithms in Fig. 5, where we normalized all the costs by the corresponding costs obtained by our approach (hence the ratios of our results are always 1). We also list the corresponding communication costs for the two data-sets in Table 2&3 separately (due to space limit, we only show the average costs over $k$ in Table 2). For the first data-set YearPredictionMSD without considering outliers (Fig. 5(a)), the clustering costs are roughly the same when $k$ is small, but our approach outperforms most of them when $k$ increases to 50 and 100 (the only one beats ours is RP-10 which has about 10 times communication cost of ours as shown in Table 2). Further, for the same data-set but considering outliers (Fig. 5(b)), the advantage of our approach is much more significant which indicates that our algorithm is more efficient for handling outliers. For the second data-set Bag of Words(NYTimes)

with a much higher dimension (Fig. 5(c)), our algorithm also outperforms the random projection based approach on both clustering cost and communication cost in most cases.

7. The bit complexity

As mentioned in Section 1.1, we assume that sending one single number costs a communication complexity of 1. In terms of the bit complexity, encoding the membership for each point needs $\log k$ bits. Thus, the bit communication complexities of the two $(9 + \epsilon)$-approximation algorithms by (Cohen et al., 2015) and us should be $O(T \log k nL + kdL) = Tn \log k + kdL$, respectively, where $L$ is the word length in the machine. We note that even in this case, our improvement is significant.

8. Future Work

Following our approach, several interesting problems deserve to be explored. For example, how to achieve a similar approximation ratio and communication complexity for $k$-median/center clustering problems. To generalize our method to the setting of arbitrary partition (such as (Kanann et al., 2014) for PCA) is also challenging but useful in some scenarios.

---

5Considering the size of Bag of Words(NYTimes), the SVD based approaches are much more impractical than the random projection based approach via current techniques in reality.
Acknowledgements

Ding was supported by a start-up fund from Michigan State University. Part of the work was done when Ding was in IIIS, Tsinghua University and Simons Institute, UC Berkeley. Liu, Huang, and Li were supported in part by the National Basic Research Program of China grants 2015CB358700, 2011CBA00300, 2011CBA00301, and the National NSFC grants 61033001, 61361136003.

References

Abadi, Daniel, Boncz, Peter, and Harizopoulos, Stavros. The design and implementation of modern column-oriented database systems. 2013.


**K-Means Clustering with Distributed Dimensions**


Jagannathan, Geetha and Wright, Rebecca N. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD ’05*, pp. 593–599, 2005.


