A Divide-and-Conquer Framework for Large-Scale Subspace Clustering

Chong You, Claire Donnat, Daniel P. Robinson, René Vidal
Johns Hopkins University, Baltimore, MD, 21218, USA

Abstract— **Given data that lies in a union of low-dimensional subspaces, the problem of subspace clustering aims to learn—in an unsupervised manner—the membership of the data to their respective subspaces. State-of-the-art subspace clustering methods typically adopt a two-step procedure. In the first step, an affinity measure among data points is constructed, usually by exploiting some form of *data self-representation*. In the second step, spectral clustering is applied to the affinity measure to find the membership of the data to their respective subspaces. While such methods are broadly applicable to mid-size datasets with $10{,}000$ data points in $10{,}000$ variables, they cannot be directly applied to large-scale datasets. This paper proposes a divide-and-conquer framework for large-scale subspace clustering. The data is first divided into chunks and subspace clustering is applied to each chunk. After removing potential outliers from each cluster, a new *cross-representation* measure for the similarity between subspaces is used to merge clusters from different chunks that correspond to the same subspace. A self-representation method is then used to assign outliers to clusters. We evaluate the proposed strategy on synthetic large-scale dataset with $1{,}000{,}000$ data points, as well as on the MNIST database, which contains $70{,}000$ images of handwritten digits. The numerical results highlight the scalability of our approach.**

I. Introduction

Recently there has been an explosion in the availability of data in many fields. In computer vision, for example, with over 300 hours of video being uploaded to YouTube every minute, it is easier to have access to databases of hundreds of terabytes. Many traditional techniques for data processing become inefficient and ineffective for data of such scale. The need for fast and efficient algorithms that are capable of processing such datasets is hence of paramount importance.

Subspace clustering. In many practical applications the intrinsic dimension of a dataset is much smaller than the dimension of the ambient space. For example, the trajectories of image points from a moving rigid object lie approximately in an affine subspace of dimension one, two, or three. Thus, the key to analyzing high-dimensional data is to find efficient and robust ways for learning its low-dimensional structure. While conventional techniques, such as Principal Component Analysis (PCA), assume the data comes from a single low-dimensional subspace, in practice data often comes from multiple classes and is hence better approximated by a union of low-dimensional subspaces. Subspace clustering is the problem of simultaneously finding the union of subspaces and the segmentation of the data into its respective subspaces.

Sparse subspace clustering. In the past decade, many subspace clustering methods ranging from statistical to algebraic ones have been studied [14]. Among them, spectral clustering based methods have been shown to be very effective for many applications. Here, we focus on the Sparse Subspace Clustering (SSC) algorithm [4], which has been shown to be robust to noise and outliers, and also enjoys various theoretical guarantees of correctness. SSC is based on the observation that a point in a low-dimensional subspace can be written as a *sparse* linear combination of other points from the same subspace. Mathematically, consider the data matrix $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ whose columns are assumed to lie in a union of low-dimensional subspaces. Given any point $\boldsymbol{x}_j$, there exists a sparse vector $\boldsymbol{c}_j$ such that $\boldsymbol{x}_j = X\boldsymbol{c}_j$ and the nonzero entries of $\boldsymbol{c}_j$ correspond to columns in X from the same subspace as $\boldsymbol{x}_j$. Such a representation $\boldsymbol{c}_j$ is called *subspace-preserving*. In principle, a subspace-preserving representation can be found by solving the optimization problem

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{c}_j\|_0 \quad \text{s.t.} \quad \boldsymbol{x}_j = X\boldsymbol{c}_j, \quad c_{jj} = 0, \tag{1}$$

where $\|\boldsymbol{c}_j\|_0$ counts the number of nonzero entries in the vector $\boldsymbol{c}_j$. It has been shown in [21] that the solution to (1) is subspace-preserving almost surely when the data on each subspace are drawn from any continuous distribution. Given the representation matrix $C = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N]$, the clustering of data X is obtained by applying spectral clustering [15] to the symmetric data affinity matrix $|C| + |C^\top|$.

Since the problem in (1) is NP-hard, the work [4] proposed to use the ℓ_1-norm instead of the ℓ_0 semi-norm, resulting in a convex optimization problem known as basis pursuit (BP):

$$\min_{\boldsymbol{c}_j} \|\boldsymbol{c}_j\|_1 \quad \text{s.t.} \quad \boldsymbol{x}_j = X\boldsymbol{c}_j, \quad c_{jj} = 0. \tag{2}$$

We will refer to the subspace clustering method based on (2) as SSC-BP. Remarkably, the solution of (2) is still subspace-preserving when the subspaces are independent [4] or satisfy certain separation conditions [5], [12]. Similar results exist for data corrupted by noise [17], corrupted by outliers [12], or pre-processed using dimension-reduction techniques [16].

Large-scale subspace clustering. Although problem (2), which is the basis for SSC-BP, can be solved in polynomial time, the SSC-BP algorithm becomes rather slow for datasets of more than 10,000 points. This has motivated many strategies for scaling-up SSC-BP. In particular, the orthogonal matching pursuit (OMP) algorithm [3], [23] is a greedy procedure for finding the sparse representation in (1), which can be used instead of (2). This method, which is called SSC-OMP in [23], was shown to be orders of magnitude faster than SSC-BP, and therefore capable

This work was supported by NSF grant 1447822.

of handling 100,000 data points. It was also shown that SSC-OMP gives a subspace-preserving representation under conditions that are comparable to those of SSC-BP [24].

Contributions. Although SSC-OMP has been shown to be more efficient than SSC-BP, it still has a computational complexity of $O(N^2)$ and thus cannot deal with datasets containing 1M points or more. The goal of this work is to develop an algorithm that is able to cluster 1M data points in a reasonable amount of time. To achieve this goal, we design a divide-and-conquer approach in which the original data is split into chunks of moderate size so that points in each chunk can be efficiently clustered using traditional subspace clustering methods. The clusters from different chunks that correspond to the same subspace are then merged to obtain a complete clustering of the original data. For this purpose, we design a novel subspace similarity measure that exploits the subspace self-expressiveness property. Unlike prior subspace similarity measures based on principal angles, the proposed measure does not require prior knowledge of the subspace dimensions, which can be hard to estimate. We also study strategies to clean the clusters, which can improve clustering accuracy. Finally, tests on both synthetic and real world datasets validate the efficiency of this approach.

II. Related work

Several scalable subspace clustering methods have been studied in the past few years, including methods based on active sets [22], truncated SVD [8], [19], factorization [11] and subsampling [10], [1]. The work of [22] proposes an active set algorithm for solving SSC-BP that is significantly faster than previous solvers and is able to cluster 70,000 data points in less than one hour. However, the algorithm is still not scalable to 1M data points as it takes more than 24 hours to cluster 581,012 data points. The works of [8] and [19] present fast algorithms for low rank representation (LRR) [9], a subspace clustering method that finds a subspace-preserving representation by learning a matrix of coefficients C that is of low-rank. They exploit the fact that the optimal solution C is of low-rank by using a truncated SVD to update C at each iteration, which reduces the computational complexity from $O(N^3)$ to $O(N^2)$. However, since the representation matrix of LRR is non-sparse and requires $O(N^2)$ memory, LRR-based methods cannot be directly applied to a dataset of size, say, 100,000 data points, as it would require $\sim$ 80GB memory. To address the memory issue, [11] exploits the fact that the representation matrix in LRR is low rank and uses a factored form of the representation matrix C to save memory. However, there are no theoretical guarantees that the method in [11] will give the correct clustering.

Subsampling-based methods have also been proposed to help scale existing methods. [10] presented the Scalable SSC method, in which SSC is applied to a subset of the data drawn at random from the entire dataset. Once the subsampled data is clustered, the remaining data points are classified to one of the computed clusters. While this method is computationally efficient, its clustering accuracy becomes sensitive to the subsampled data, i.e., it requires the subsampled data to well represent the distribution of points in all subspaces. The work of [1] learns a small-sized dictionary and clusters the data based on the affinity between the data points and the dictionary atoms. While the learned dictionary would be expected to be more representative of the data than the random subsampled data used by Scalable SSC, there are no theoretical guarantees on the quality of the dictionary for the purpose of clustering. Indeed, the clustering accuracy is reduced for many cases in the empirical evaluation in [1].

III. A divide-and-conquer approach

We propose a divide-and-conquer approach to large-scale subspace clustering called SSC-DC. The entire procedure (see Algorithm 1) consists of four phases. First, instead of learning a representation matrix for the entire data, SSC-DC randomly splits the data into smaller chunks and then independently performs SSC on each chunk. Second, since the resulting clusters may contain points from more than one subspace, SSC-DC uses an outlier pursuit procedure to separate inliers from outliers within each cluster. Third, a new similarity measure between clusters is used to merge clusters of inliers that correspond to the same subspace. Finally, once the clusters have been merged, the outliers are reclustered by assigning them to one of the merged clusters.

A. *Phase 1: split and cluster*

First, SSC-DC partitions the N data points into B disjoint chunks, $\{X^{(b)}\}_{b=1}^{B}$, where the size of each chunk N/B (we assume that B divides N) is small enough so that the chunks can be handled by modern SSC methods. Once the chunks have been formed, SSC-BP or SSC-OMP is applied to each chunk $X^{(b)}$ to get n clusters $\{X_\ell^{(b)}\}_{\ell=1}^{n}$, where n is the number of subspaces and is set to be the same for all chunks.

B. *Phase 2: detect outliers*

If the clustering within each chunk from Phase 1 were perfect, then each cluster would only contain points from a single subspace. In practice, some clusters could be corrupted by points from other subspaces, which could significantly affect the ability to merge the clusters in Phase 3.

To address this issue, we apply an outlier detection algorithm to each of the clusters to identify and remove outliers. Specifically, assume that each one of the clusters $\{X_\ell^{(b)}\}_{\ell\in\{1,\cdots,n\}}^{b\in\{1,\cdots,B\}}$ obtained in Phase 1 contains many points from one of the subspaces as well as a few points from other subspaces. The goal of Phase 2 is to detect and remove points from other subspaces (a.k.a. outliers), thus generating a submatrix $\bar{X}_\ell^{(b)}$ of the matrix $X_\ell^{(b)}$ for all $\ell \in \{1, \ldots, n\}$ and $b \in \{1, \ldots, B\}$ that contains only points from one of the subspaces (a.k.a. inliers). The outlier detection problem has been studied in the context of robust PCA, e.g. see [20], [7], [13]. In this work, we use the Outlier Pursuit method of [20], which aims to decompose the data matrix as $X_\ell^{(b)} = L + S$. Here, L is some low-rank matrix whose non-zero columns (the inliers) span the underlying subspace containing $X_\ell^{(b)}$, and S is a column-sparse matrix (i.e. there are only a few nonzero columns) whose non-zero columns correspond to the outliers. To compute L and S, one solves

$$\min_{L,S} \|L\|_* + \lambda \|S\|_{2,1} \quad \text{s.t.} \quad X_\ell^{(b)} = L + S, \tag{3}$$

where $\lambda > 0$ is a trade-off parameter, $\|L\|_*$ is the nuclear norm of L defined as the sum of the singular values of L, and $\|S\|_{2,1}$ is the sum of the ℓ_2-norms of the columns of S. Once the outliers have been detected as the non-zero columns of S, we assign them to an outlier set to be processed in Phase 4.

C. Phase 3: merge subspaces

Given the data matrices $\{\bar{X}_\ell^{(b)}\}_{\ell\in\{1,\cdots,n\}}^{b\in\{1,\cdots,B\}}$, each one containing ideally data from only one subspace, the goal of Phase 3 is to merge clusters whose data come from the same subspace. For this purpose, we adopt a two-step procedure in which pairwise similarities between clusters are computed and spectral clustering is applied to the resulting similarity.

A classical measure of the similarity between two subspaces is their principal angle, which can be computed from the largest singular value of the matrix $U^\top V$, where U and V are orthogonal bases for the two subspaces. However, since real data typically contains noise, it can be difficult to compute a basis for each subspace since the dimensions of the subspaces are unknown and nontrivial to estimate. Given two data submatrices whose columns are from the same subspace, either overestimation or underestimation of the subspace dimension can result in an inaccurate estimation of the principal angle. Thus, there is a need to design measures of subspace similarity that do not require an estimate of the subspace dimension and are robust to noisy data.

Motivated by the fact that points in the same subspace can be used to mutually express each other, we design a "cross-expressiveness" based similarity measure for two subspaces. The idea is that if the columns of two matrices Y_1 and Y_2 are drawn from the same subspace, then it holds that $Y_1 = Y_2 \cdot C_1$ for some C_1, i.e., each column of Y_1 can be expressed using the columns from Y_2. On the other hand, if Y_1 and Y_2 are drawn from two different subspaces (assume that one subspace does not contain the other), then such a representation C_1 does not exist because the columns of Y_1 cannot be expressed using columns from Y_2. Motivated by this ideal setting, and to cope with the possible existence of noise in the data, we compute the representation C_1 as:

$$C_1 = \operatorname*{argmin}_C \|Y_1 - Y_2 C\|_F^2 + \lambda \|C\|_F^2, \tag{4}$$

for some weighting parameter $\lambda > 0$. Note that problem (4) has the closed form solution $C_1 = (Y_2^\top Y_2 + \lambda I)^{-1} Y_2^\top Y_1$. Our new dissimilarity measure between Y_1 and Y_2 is then defined as

$$d(Y_1, Y_2) = \frac{1}{2}\left(\frac{\|Y_1 - Y_2 C_1\|_F}{\|Y_1\|_F} + \frac{\|Y_2 - Y_1 C_2\|_F}{\|Y_2\|_F}\right), \tag{5}$$

where C_1 is computed from (4) and C_2 is computed by swapping Y_1 and Y_2 in (4). Based on the dissimilarity measure (5), the similarity between Y_1 and Y_2 is computed as $\exp(-d(Y_1, Y_2)/(2\sigma^2))$ for some parameter $\sigma > 0$.

Algorithm 1 SSC Divide-and-conquer

```
Input: The data matrix $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$, the number of clusters $n$, and the number of chunks $B$.
1: Phase 1: Split and cluster
     Split the data matrix $X$ evenly into $\{X^{(b)}\}^{b=1,\cdots,B}$.
     Apply SSC on each $X^{(b)}$ to get clusters $\{X_\ell^{(b)}\}_{\ell=1,\cdots,n}^{b=1,\cdots,B}$.
2: Phase 2: Detect outliers
     For each matrix in $\{X_\ell^{(b)}\}_{\ell=1,\cdots,n}^{b=1,\cdots,B}$ solve (3). Place points corresponding to nonzero columns of $S$ into the outlier set, and denote the matrix containing the remaining points as $\bar{X}_\ell^{(b)}$.
3: Phase 3: Merge subspaces
     Compute similarities $\exp(-d(\bar{X}_\ell^{(b)}, \bar{X}_{\ell'}^{(b')})/(2\sigma^2))$ using (5) for all pairs $(b, \ell) \neq (b', \ell')$. Apply spectral clustering using this similarity matrix to get clusters $\{\bar{X}_\ell\}_{\ell=1,\cdots,n}$.
4: Phase 4: Recluster outliers
     Assign each $\boldsymbol{y}$ in the outlier set to one of the clusters in $\{\bar{X}_\ell\}_{\ell=1,\cdots,n}$ by using (7).
Output: A clustering of $X$ into $n$ clusters $\{\bar{X}_\ell\}_{\ell=1,\cdots,n}$.
```

D. Phase 4: recluster outliers

After the merging procedure in Phase 3, the algorithm has generated n clusters $\{\bar{X}_\ell\}_{\ell=1,\cdots,n}$, each of which will ideally contain only points from one of the ground-truth subspaces. However, the points that were detected as outliers in Phase 2 still need to be assigned to one of the clusters. A simple approach for assigning outliers to clusters would be to fit a subspace to each one of the clusters that have already been obtained and then assign each outlier to one of those n clusters. However, this would require us to know the dimension of the subspaces, and any errors in the estimation of the dimensions could lead to errors in the assignments.

To address this issue, we note that since the class labels for clusters $\{\bar{X}_\ell\}_{\ell=1}^n$ have already been generated, we can treat $\{\bar{X}_\ell\}_{\ell=1}^n$ as training data and use any supervised classification technique to classify each point in the outlier set. Here, we adopt a representation based classification technique [18], [25]. If we define $\bar{X} = [\bar{X}_1, \cdots, \bar{X}_n]$, then a representation for any outlier point $\boldsymbol{y}$ may be found by solving[1]

$$\min_{\boldsymbol{c}} \|\boldsymbol{y} - \bar{X}\boldsymbol{c}\|_2^2 + \lambda \|\boldsymbol{c}\|_2^2 \tag{6}$$

for some parameter $\lambda > 0$. Ideally, the nonzero entries in the representation vector $\boldsymbol{c}$ will correspond to points in $\bar{X}$ that are from the same subspace as $\boldsymbol{y}$. In practice, nonzero entries of $\boldsymbol{c}$ may be distributed among multiple subspaces. Following the procedure in [18], let $\delta_\ell(\boldsymbol{c})$ be a vector of the same size as $\boldsymbol{c}$ such that the entries of $\delta_\ell(\boldsymbol{c})$ are all zero except for those that correspond to $\bar{X}_\ell$, which are equal to the corresponding entries of $\boldsymbol{c}$. The point $\boldsymbol{y}$ is then assigned to the class ℓ that gives the minimum representation residual

[1]Observe that this problem is potentially huge when D and N are large. However, when D is small one can use the inversion lemma to solve for $\boldsymbol{c}$ efficiently. As we shall see in our experiments, the time spent in Phase 4 is not significant in practice. That being said, simpler classification methods can be tried when D and N are both large.

of $\boldsymbol{y}$ using $\delta_\ell(\boldsymbol{c})$, i.e., to the class ℓ that solves

$$\min_{\ell} \|\boldsymbol{y} - \bar{X}\delta_\ell(\boldsymbol{c})\|_2. \tag{7}$$

This completes the description of our SSC-DC framework.

IV. Experiments

A. Synthetic data

To test the effectiveness of SSC-DC on large-scale datasets and to study how the number of chunks affects the running time, we design experiments using synthetic data. First, we choose 10 subspaces each of dimension 5, independently and uniformly at random in an ambient space of dimension 15. Second, we generate an equal number of data points uniformly at random on each of the subspaces. The total number of data points is varied from 10,000 to 1,000,000.

For Phase 1 of SSC-DC, the data is randomly divided into 10, 50 or 100 chunks and SSC-BP is applied to each chunk. We use the SPAMS package[2] to solve the sparse representation problem (2) used by SSC-BP. The clusters are merged according to Phase 3 of Algorithm 1 with $\lambda = 0.1$ in (4) and $\sigma = 1$. We do not perform outlier detection (i.e. Phase 2) and reclustering (i.e. Phase 4) in this experiment. The running times of different algorithms on different number of data points are shown in Figure 1.

The curve for SSC is the baseline, which was obtained by applying SSC-BP to the entire data, i.e., it uses one chunk. We see that while it only needs one minute to handle 10,000 data points, it takes around 14 hours to cluster $\sim$360,000 data points. Since the running time for each algorithm is limited to 24 hours, SSC does not finish clustering 1,000,000 data points. If we use SSC-DC with 10 chunks, we can observe a significant reduction on the running time for all scales of the tested data. In particular, SSC-DC with 10 chunks uses around one hour to cluster $\sim$360,000 in comparison to the 14 hours used by SSC. By using the divide-and-conquer strategy, our method is able to cluster 1,000,000 data points.

When the number of chunks in SSC-DC is increased from 10 to 50, we observe that the running time increases when the data size is relatively small, but decreases when the data size is relatively large. This illustrates the trade-off in choosing the number of chunks in SSC-DC. Although increasing the number of chunks reduces the scale of the problems solved by SSC on each chunk, it also increases the number of subspaces to be merged in Phase 3. Therefore, the computational cost associated with these two effects should be balanced in practice. As shown in Figure 1, SSC-DC (100 chunks) is less efficient than SSC-DC (50 chunks), although it is likely that SSC-DC (100 chunks) will be more efficient that SSC-DC (50 chunks) when the dataset size grows beyond 1,000,000 data points.

B. MNIST handwritten digit data

To evaluate the performance of SSC-DC on real data, we use the MNIST handwritten digit database [6]. It contains 70,000 images of handwritten digits 0-9. Following the setup

[2]http://spams-devel.gforge.inria.fr/

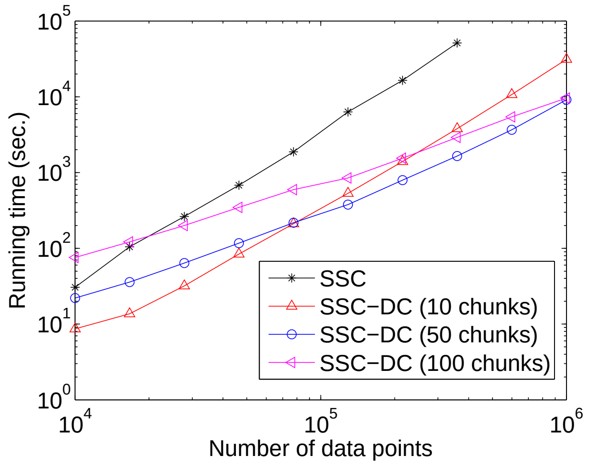

Fig. 1. Performance of SSC-DC with different numbers of chunks on synthetic data. We randomly generate 10 subspaces of dimension 5 in an ambient space of dimension 15, and then randomly draw the same number of points from each subspace. The x-axis gives the overall number of points, which is varied from 10,000 to 1,000,000 and shown in log scale. The y-axis reports the running time in log scale. The missing points for SSC indicate that the algorithm does not finish within 24 hours.

in [22], [23], we compute the features for each image from a scattering transform network [2]. The feature vectors are of size 3,472, but are then projected to dimension 500 using PCA, resulting in the data to be clustered.

We test the capability of SSC-DC on clustering the entire 70,000 images into 10 groups corresponding to digits 0-9. We use SSC-OMP for subspace clustering in Phase 1. Note that the solution for S in (3), which is supposed to be column sparse, contains dense noise because the dataset contains noise. Therefore, we compute the ℓ_2-norm of the columns of S, and declare those that are larger than a threshold as outliers; we found it difficult to determine a proper threshold for declaring outliers. For simplicity, we sort the ℓ_2-norms of the columns of S in descending order, and declare the data that correspond to the first 10% as outliers. While this may produce many false outliers, it is unlikely to significantly affect the final clustering result because the false outliers are reclustered in Phase 4.

Table I shows clustering accuracy and running time for various number of chunks. The results are averages over 10 independent trials. In terms of clustering accuracy, the performance of SSC-DC becomes worse as the number of chunks increases. This is in accordance with the analysis and empirical results in [23], where it is shown that the clustering performance of SSC-OMP becomes better when there are more samples in each subspace. Therefore, the clustering on each chunk in Phase 1 becomes less accurate as the number of chunks increases, which affects the final clustering accuracy. Table I also reports the performance of SSC-OMP, which exactly corresponds to the same computation in Phase 1 of SSC-DC (1 chunk). Note that the clustering accuracy of SSC-OMP is considerably lower than SSC-DC (1 chunk), which demonstrates that the outlier detection and reclassification procedures in Phase 2 and Phase 4 of SSC-

TABLE I
PERFORMANCE OF DIFFERENT SUBSPACE CLUSTERING METHODS ON THE MNIST DATASET. THE DATASET CONSISTS OF 70,000 IMAGES OF 10 HANDWRITTEN DIGITS (I.E. , 0-9). SSC-DC WITH NUMBER OF CHUNKS $B \in \{1, 2, 5, 10, 20, 50, 100, 200\}$, AS WELL AS OTHER METHODS FOR COMPARISON, ARE APPLIED TO FEATURES EXTRACTED FROM THE IMAGES BY A SCATTERING NETWORK, WHICH ARE THEN PROJECTED TO DIMENSION 500 USING PCA.

Method	Acc. (%)	Time (sec.)				
		Total	P1	P2	P3	P4
SSC-DC (1)	96.55	5254	1825	3304	30	93
SSC-DC (2)	96.10	4390	1049	3185	59	94
SSC-DC (5)	94.90	1596	436	937	134	88
SSC-DC (10)	93.04	1081	272	454	266	88
SSC-DC (20)	91.46	1081	196	274	523	87
SSC-DC (50)	89.07	1689	169	183	1243	93
SSC-DC (100)	85.46	2635	148	132	2260	94
SSC-DC (200)	78.93	5518	144	119	5161	93
SSC-OMP	94.75	1825	NA	NA	NA	NA
SSC-BP	92.46	80987	NA	NA	NA	NA
EnSC [22]	93.79	1694	NA	NA	NA	NA
OLRSC [11]	75.30	1284	NA	NA	NA	NA

DC effectively boost clustering accuracy.

The third column of Table I reports the overall running time of SSC-DC. It shows that the running time first decreases as the number of chunks increases, and then starts increasing once the number of chunks is larger than 20. This behavior can be explained by the breakdown of the running time for the 4 phases as reported in columns 4–7 of Table I. In particular, notice that the running time of Phase 1 strictly decreases as the number of chunks increases, showing that SSC becomes more efficient when the chunks are smaller. On the other hand, the running time of Phase 3 strictly increases as the number of chunks increases, showing that it quickly becomes prohibitively expensive to merge subspaces as the number of subspace increases. At last, we note that Phase 2 (outlier detection via Outlier Pursuit) is the bottleneck in running time when the number of chunks is small. Thus, SSC-DC has the potential to be more efficient by using more scalable algorithms for outlier detection.

For comparison purposes, the results for several methods are reported at the bottom of Table I. We can see that SSC-BP is more than one order of magnitude slower than the other methods. The EnSC [22] and the OLRSC [11], which use an active-set approach and factorization scheme, respectively, are able to efficiently cluster the data. However, EnSC is still slower when compared with SSC-DC (5 chunks), and OLRSC does not have comparable clustering accuracy.

V. CONCLUSION

We presented a divide-and-conquer strategy for subspace clustering aimed at large-scale datasets. We designed a new measure of subspace similarity that did not require the knowledge of the subspace dimension and that is effective in merging subspaces obtained from subspace clustering performed on each chunk of data. We also adopted an outlier detection and reclustering procedure to clean computed clusters and thus improve clustering accuracy. Our framework was able to cluster 1,000,000 data points in approximately the time required by other modern SSC methods to cluster about 100,000 data points.

REFERENCES

[1] A. Adler, M. Elad, and Y. Hel-Or. Linear-time subspace clustering via bipartite graph modeling. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2234 – 2246, 2015.
[2] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.
[3] E. L. Dyer, A. C. Sankaranarayanan, and R. G. Baraniuk. Greedy feature selection for subspace clustering. *Journal of Machine Learning Research*, 14(1):2487–2517, 2013.
[4] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.
[5] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.
[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2324, 1998.
[7] G. Lerman, M. B. McCoy, J. A. Tropp, and T. Zhang. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics*, 15(2):363–410, 2015.
[8] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low rank representation. In *Neural Information Processing Systems*, 2011.
[9] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning*, pages 663–670, 2010.
[10] X. Peng, L. Zhang, and Z. Yi. Scalable sparse subspace clustering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2013.
[11] J. Shen, P. Li, and H. Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 622–631, 2016.
[12] M. Soltanolkotabi and E. J. Candès. A geometric analysis of subspace clustering with outliers. *Annals of Statistics*, 40(4):2195–2238, 2012.
[13] M. Tsakiris and R. Vidal. Dual principal component pursuit. In *ICCV Workshop on Robust Subspace Learning and Computer Vision*, pages 10–18, 2015.
[14] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(3):52–68, March 2011.
[15] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17, 2007.
[16] Y. Wang, Y. Wang, and A. Singh. A deterministic analysis of noisy sparse subspace clustering for dimensionality-reduced data. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1422–1431, 2015.
[17] Y.-X. Wang and H. Xu. Noisy sparse subspace clustering. *Journal of Machine Learning Research*, 17(12):1–41, 2016.
[18] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, 2009.
[19] S. Xiao, W. Li, D. Xu, and D. Tao. Falrr: A fast low rank representation solver. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4612–4620, 2015.
[20] H. Xu, C. Caramanis, and S. Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.
[21] Y. Yang, J. Feng, N. Jojic, J. Yang, and T. S. Huang. ℓ_0-sparse subspace clustering. In *European Conference on Computer Vision*, pages 731–747. Springer, 2016.
[22] C. You, C.-G. Li, D. Robinson, and R. Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
[23] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
[24] C. You and R. Vidal. Geometric conditions for subspace-sparse recovery. In *International Conference on Machine learning*, pages 1585–1593, 2015.
[25] L. Zhang, M. Yang, and X. Feng. Sparse representation or collaborative representation: Which helps face recognition? In *IEEE International Conference on Computer Vision*, pages 471–478, 2011.