Local-Density Subspace Distributed Clustering for High-Dimensional Data

Yangli-ao Geng, Qingyong Li, *Member, IEEE*, Mingfei Liang, Chong-Yung Chi, *Fellow, IEEE*, Juan Tan, and Heng Huang

Abstract—Distributed clustering is emerging along with the advent of the era of big data. However, most existing established distributed clustering methods focus on problems caused by a large amount of data rather than caused by the large dimension of data. Consequently, they suffer the "curse" of dimensionality (e.g., poor performance and heavy network overhead) when high-dimensional (HD) data are clustered. In this article, we propose a distributed algorithm, referred to as Local Density Subspace Distributed Clustering (LDSDC) algorithm, to cluster large-scale HD data, motivated by the idea that a local dense region of a HD dataset is usually distributed in a low-dimensional (LD) subspace. LDSDC follows a local-global-local processing structure, including grouping of local dense regions (atom clusters) followed by subspace Gaussian model (SGM) fitting (flexible and scalable to data dimension) at each sub-site, merging of atom clusters at every sub-site according to the merging result broadcast from the global site. Moreover, we propose a fast method to estimate the parameters of SGM for HD data, together with its convergence proof. We evaluate LDSDC on both synthetic and real datasets and compare it with four state-of-the-art methods. The experimental results demonstrate that the proposed LDSDC yields best overall performance.

Index Terms—High-dimensional clustering, distributed clustering, density-base clustering, subspace Gaussian model

1 Introduction

Clustering, as a branch of unsupervised learning, is a process of discovery and exploration for investigating inherent and hidden structures within a large dataset [1]. To be specific, clustering is to partition a set of data samples (or observations) into subsets. Each subset is a cluster, such that samples in a cluster are similar to one another, yet dissimilar to samples in other clusters. Clustering has been extensively applied to a variety of data-analysis tasks [2], [3]. However, two main challenges are faced by centralized clustering algorithms when handling big data. First, they require that all data are collected in a central machine, which may be prohibited in the practical scenario where a large amount of heterogeneous, complex data reside on different, distributed working computers which are connected to each other via local or wide area networks [4]. Second, large size and high dimension of big data may prohibit any clustering algorithm from operating in a single machine due to efficiency and cost considerations.

To meet these challenges, many clustering methods have been proposed to process big data in a distributed system. Existing distributed clustering algorithms can be basically divided into four groups: partition-based methods [5], grid-based methods [6], density-based methods [7] and model-based methods [7]. Partition-based methods employ an iterative strategy to divide data into a certain of clusters according to some predefined quantitative optimization criteria. Grid-based methods quantize the original data space into a finite number of grids, and then group the grids according to statistical characteristics of samples in each grid. Density-based methods first estimate the distribution density of objects in a feature space, and then search for high density regions (i.e., clusters) separated by regions of lower density. Model-based methods first optimize the fitting of local data for some (predefined) parametric statistical models, and then the estimated parameters are collected and used in a central machine to form clusters. There is no doubt that distributed clustering has become a fundamental tool for high-level big data analysis.

Recently, clustering of high-dimensional (HD) data in a distributed environment is emerging due to the rapid growth of data volume and the flourishing of distributed computing technologies, with possible applications including text grouping [8], image retrieval [9] and hyperspectral image analysis [10]. However, HD data involve several issues to distributed clustering methods. The first one is that the shapes of clusters become more complex as the dimension increases. Various shapes may bring about an intractable challenge to partition-based methods, which perform well mainly for convex clusters

- *Y.-A. Geng and Q. Li are with the Beijing Key Lab of Transportation Data Analysis and Mining, Beijing Jiaotong University, Beijing 100044, China. E-mail: {gengyla, liqy}@bjtu.edu.cn.*
- *M. Liang is with WeiXin Group, Tencent Company Limited, Beijing 100044, China. E-mail: mingfeiliang@bjtu.edu.cn.*
- *C.-Y. Chi is with Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan. E-mail: cychi@ee.nthu.edu.tw.*
- *J. Tan is with the Department of Business Administration, Beijing Technology and Business University, Beijing 100048, China. E-mail: tanjuan@btbu.edu.cn.*
- *H. Huang is with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15260, and is also with JD Finance America Corporation. E-mail: heng.huang@pitt.edu.*

Manuscript received 10 Sept. 2019; revised 9 Jan. 2020; accepted 9 Feb. 2020. Date of publication 24 Feb. 2020; date of current version 23 Mar. 2020. (Corresponding authors: Juan Tan and Qingyong Li.) Recommended for acceptance by Q. Zheng. Digital Object Identifier no. 10.1109/TPDS.2020.2975550

[11]. The second one is that the number of possible grids grows exponentially with the number of dimensions. This exponential increase may impose heavy overheads on both computation and transmission of clustering information for grid-based methods [12]. The third one is that data samples become increasingly sparse as the dimension increases, thereby inflating the variance of estimated density and thus yielding a performance degradation to density-based algorithms [11]. The last one is that the parameter estimation for HD models is computation consuming. As a result, model-based methods usually need to compromise between performance accuracy and computation cost [13]. Overall, clustering HD distributed data effectively and efficiently is still an open problem.

Motivated by an empirical observation that a local dense region of a HD dataset is usually embedded in a low-dimensional (LD) subspace [14], [15], this paper proposes a novel approach for distributed clustering of HD datasets, referred to as Local-Density Subspace Distributed Clustering (LDSDC) algorithm, that operates in a local-global-local fashion. First, in each sub-site, LDSDC exploits a density-based method to divide its dataset into local regions. Second, the data in each local region is fitted by a subspace Gaussian model (SGM), which is flexible and scalable to the data dimension. Third, in the global site, LDSDC collects all the local SGMs from sub-sites, and based on them generates a global density distribution. Furthermore, local SGMs are merged according to "connection values" (to be defined in (6) below) under the global density distribution, and then the resulting merging information are broadcasted to all the sub-sites. Lastly, in each sub-site, local regions are merged into final clusters according to the received merging information. In brief, this work makes the following contributions.

1) A distributed clustering algorithm LDSDC based on SGM is proposed to handle large-scale and HD data.
2) An illustrative analysis of SGM is presented to demonstrate its superiority in high-dimensional data modeling.
3) A fast method for the estimation of SGM parameters is proposed, including an algorithm and its convergence proof.

A preliminary version of this work was published in ICPADS 2017 [16], in which a distributed algorithm named REMOLD is proposed. In this paper, it is significantly extended from both theoretical and practical aspects. Specifically, beyond the general Gaussian model utilized by REMOLD, the SGM (a compact statistical model) is considered for local cluster modeling, which proves efficient and effective in HD data clustering. Then we present a theoretical analysis to demonstrate its superiority over general Gaussian model in HD data modeling. Furthermore, a fast parameter-estimation method for the SGM is proposed. Then extensive experiments are presented to show the effectiveness of the proposed LDSDC by clustering HD data (instead of two-dimensional datasets adopted in [16]), including various synthetic datasets for different scales, dimensions and cluster shapes. Finally, we present some experimental results on real HD datasets, which further show the efficacy and practicability of the proposed LDSDC.

The rest of the paper is organized as follows. Section 2 introduces the related works and notations. Section 3 presents the proposed clustering method LDSDC. Section 4 details subspace Gaussian model. Section 5 provides some simulation results and experimental results to demonstrate the efficacy of the proposed LDSDC. Finally, we conclude the paper in Section 6.

2 Related Works and Notations

2.1 Distributed Clustering

Distributed clustering aims to handle data inherently residing on distributed sites, which cannot be collected in a central site due to practical issues such as privacy concerns and limited transmission bandwidth [17]. Unlike centralized clustering, distributed clustering relies heavily on the network structure because different networks have their own limitations and task objectives [18]. In general, distributed clustering methods are designed for two network topologies: global-sub-site networks and peer-to-peer (P2P) networks [19]. The global-sub-site paradigm demands a reliable global site to collect all necessary information from distributed sub-sites. By contrast, the P2P paradigm does not have a central server and all sites with limited view coverage of the entire network only exchange necessary information to perform their own local clustering tasks [18], [19]. In this paper, we focus on distributed clustering methods for the global-sub-site networks.

The well-known method, k-means [20] is the basis of many distributed methods. To avoid the exposure of private data in the transmission, Vaidya and Clifton [21] proposed a distributed k-means with the privacy-protection taken into consideration. It can achieve good performance when different sites contain different private attributes for a common set of entities. Distributed k-means algorithm is basically an ensemble-learning-based distributed clustering algorithm introduced in [22]. This method first does clustering in sub-site using k-means, then send all centroid values to the central site, and finally global centroid values of the underlying global clustering are obtained by using k-means again [23]. Jeong *et al.* [24] proposed a modified distributed k-means algorithm for clustering huge amount of distributed biological data. Balcan *et al.* [25] provided a distributed method for constructing a global core-set of data samples, based on which k-means clustering can handle distributed data efficiently. Overall, these methods are efficient enough to deal with big data, but their performance is on par with that of the conventional k-means method.

In [26], Xu *et al.* presented a parallel clustering method, based on the conventional centralized clustering algorithm DBSCAN [27]. First of all, it divides the dataset into several partitions at the global site, and then send each partition to a sub-site, where the partition is locally clustered by using DBSCAN. Finally, the clustering results obtained from the sub-sites are merged at the global site. In [4], a DBSCAN-based distributed clustering algorithm named DBDC was proposed. It first clusters data at each sub-site, and some specific core samples and the associated neighborhood parameters are extracted at the same time, which are then collected at a global site. Based on these core samples and parameters, DBSCAN is used to generate the final clusters at the global site. Though DBDC can efficiently find clusters with arbitrary shapes, the excessive input-parameters and high performance sensitivity to the input-parameters weaken its practical

applicability. Accordingly, approaches such as SDBDC [28] and LDBDC [29] have been proposed to improve on DBDC. On the whole, these methods can detect clusters with complex shapes in a distributed system, but they are confronted with a variety of densities of HD clusters.

Density-Peak (DP) clustering [30] is a centralized clustering algorithm that has distinctive advantages over many other algorithms. Based on DP clustering, Gong *et al.* [31] proposed an efficient distributed density peak clustering (EDDPC) algorithm. It employs Voronoi diagram and data replication/filtering that significantly reduce the amount of both useless distance measuring and data shuffle in data processing. Zhang *et al.* [32] proposed a distributed DP clustering algorithm, named LSHDDP, based on a locality sensitive hashing (LSH) function. The LSH function has the property: samples that are closer to each other have a higher collision probability. LSHDDP exploits LSH for partitioning data, performs local computation, and aggregates local results to obtain the final results. It has been shown that LSHDDP can achieve a factor of 2x speed-up over the EDDPC approach, while returning comparable clustering results. Overall, these methods can reproduce the performance of the centralized DP clustering, but they usually require transmission of a considerable amount of data samples, thus adverse to network-transmission and privacy-protection.

Kriegel *et al.* [33] proposed a distributed model-based clustering algorithm. Their method uses the expectation maximum (EM) method [34] for detecting local models in terms of mixtures of Gaussian distributions. Then, these local Gaussian distributions are merged in the global site to generate a more meaningful global model. In [35], Lin *et al.* presented a clustering technique with distributed EM mixture modeling. This method controls data sharing, and prevents disclosure of individual data attributes or any results that can be traced back to an individual site. In summary, these methods are effective and efficient for LD data, but usually they have to sacrifice accuracy to compromise the computation cost for HD data.

2.2 High-Dimensional Clustering

With regard to centralized clustering HD data, many methods utilize global dimension-reduction techniques followed by a standard clustering algorithm. The most popular dimension-reduction technique is Principal Component Analysis (PCA) [36]. PCA aims to find a linear mapping from the original space to a LD space such that the mapped data can preserve the variance of the original data. However, the distribution manifold of samples in the original feature space is sometimes non-linear. Applying PCA in this case will lose the non-linear structure information. To overcome this challenge, kernel PCA [37] utilizes a user-specified kernel (similarity matrix) to implicitly construct a non-linear mapping from the original feature space to a new space, and performs PCA in the new space via the kernel trick [38]. The effectiveness of the kernel PCA highly relies on the choice of the kernel, but lacking in feasible methods for the kernel choice. In [39], Tipping and Bishop proposed a probabilistic PCA, which gives PCA a statistical interpretation, provided that an observation sample $\mathbf{x} \in \mathbb{R}^M$ is generated by the sum of three latent terms, i.e., $\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \epsilon\mathbf{n}$, where $\mathbf{z} \sim G(\mathbf{0}, \mathbf{I}_D)$ and $\mathbf{n} \sim G(\mathbf{0}, \mathbf{I}_M)$ are two latent random variables that follow the standard Gaussian distribution. $\mathbf{W} \in \mathbb{R}^{M\times D}$, $\boldsymbol{\mu} \in \mathbb{R}^M$ and $\epsilon \in \mathbb{R}$ are unknown parameters to be estimated by the probabilistic PCA. Generalized PCA [40] is an algebraic method to handle the case where data distribute in multiple LD subspaces. It has a solid theory foundation, but extra high complexity restricts its application. Robust PCA [41] estimates the principal subspace of data via convex programming and it has shown better performance than PCA while processing data in the presence of outliers.

On the other hand, there are some clustering methods designed to process HD data directly. CLIQUE [42] is a pioneering approach to cluster HD data. It discretizes the feature space into regular intervals in every dimension and thus generates a grid structure. Then dense cells that contain more samples exceeding a threshold are combined with other dense cells in a bottom-up approach [43]. Adjacent dense cells are merged to establish the actual clusters. CLIQUE scales linearly with input size, so it has good scalability as the data dimension in the data is increased. However, obtaining a meaningful clustering is dependent on proper tuning of the grid size and the threshold. This may be difficult in practice because the grid size and threshold are used across all dimension combinations in the dataset [1]. Many extended methods [44], [45], [46] have been proposed to overcome the drawbacks of CLIQUE. By integrating density-based clustering and subspace clustering, Kailing *et al.* [47] proposed a density connected subspace clustering method which can detect arbitrarily shaped clusters in subspaces of the original HD space. Charles *et al.* [48] introduced subspace Gaussian distribution for the research area of clustering and then proposed a HD data clustering method with it. This method follows a mixture-model framework, in which the models are limited to subspace Gaussian distribution. Then an EM-based algorithm is developed to estimate the parameters of the mixture-model. Finally, each sample is assigned to a cluster following the maximum posterior probability. Their experiments show that this method outperforms other HD clustering methods.

In summary, though distributed clustering and HD clustering have been extensively studied separately, limited works focus on clustering HD data with low overhead under distributed environments, which is also a common problem in recent applications. The LDSDC proposed in this work provides a feasible solution to this problem.

For ease of the ensuing presentation, some notations are defined hereafter. Italic symbols (e.g., a and A) represent scalars. Calligraphic symbols denote sets (e.g., $\mathcal{A}$) or subspaces (e.g., $\mathcal{S}$). Boldface capital (e.g., $\mathbf{A}$) and boldface lowercase (e.g., $\mathbf{a}$) symbols denote matrices and vectors, respectively. $\mathbf{I}_M$ and 1_M represent the identity matrix in $\mathbb{R}^{M\times M}$ and the all-one vector in $\mathbb{R}^M$, respectively. $\lambda_i(\mathbf{A})$ denotes the ith largest eigenvalue of a symmetric matrix $\mathbf{A}$. $\oplus_{i=1}^{I} \diamond_i \triangleq \mathrm{diag}(\diamond_1, \ldots, \diamond_I)$ is a block diagonal matrix, where the ith diagonal entry $\diamond_i$ can be a matrix or a scalar. $\|\mathbf{a}\|$ represents the ℓ_2-norm of $\mathbf{a}$. We use $\mathrm{Tr}(\mathbf{A})$, $\mathcal{R}(\mathbf{A})$ and $\|\mathbf{A}\|_F$ to represent respectively the trace, the range space and the Frobenius norm of $\mathbf{A}$. $|\cdot|$ denotes the cardinality of a set or the determinant of a square matrix.

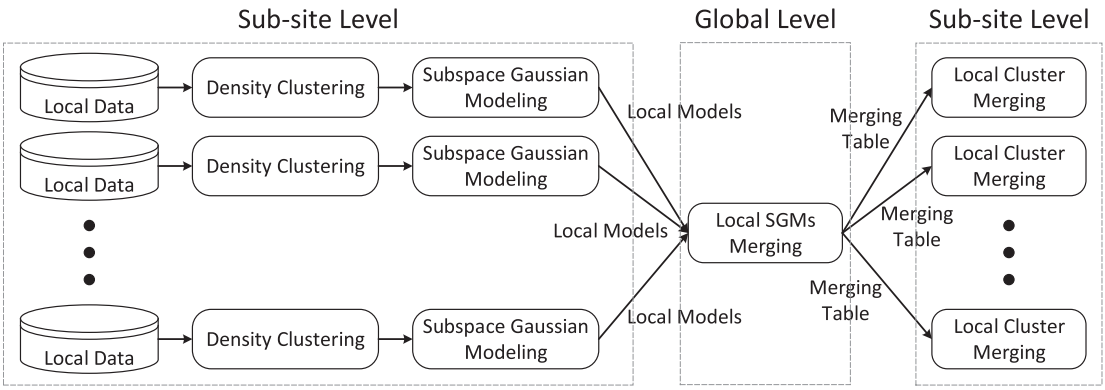


Fig. 1. The structure of the proposed LDSDC algorithm for distributed clustering.

3 Local-Density Subspace Distributed Clustering

In this section, we present the LDSDC algorithm. Consider a scenario that consists of a global site and W sub-sites. The total sample set $\mathcal{X} = \bigcup_{w=1}^{W} \mathcal{X}_w$ is distributed and stored in W sub-sites with $\mathcal{X}_w = \{\mathbf{x}_n^w \in \mathbb{R}^M\}_{n=1}^{N_w}$ stored in the wth sub-site. Our goal is to cluster $\mathcal{X}$ with low overhead in this distributed system. As shown in Fig. 1, our algorithm comprises four stages, as follows:

1) In each sub-site, LDSDC divides samples into local clusters using a density-based method, where each local cluster is termed an atom cluster.
2) In each sub-site, LDSDC models each atom cluster using a SGM. Then, estimated SGM parameters of all models are sent to the global site.
3) In the global site, a global density distribution is generated by integrating all local parameters. Based on the global density distribution, LDSDC calculates the connection values (defined in (6) below) for all the pairs of local models, and then merge similar models according to the connection values. Then the obtained merging result is sent back to each sub-site.
4) In each sub-site, similar atom clusters are merged according to the received merging result.

However, when the given dataset is ill-distributed in different sub-sites (e.g., samples stored in sub-sites are unbalanced, and/or many dense clusters are partitioned in different sub-sites). In this case, prior to applying the 4-stage LDSDC, data reshuffling according to Local Sensitive Hashing values [49] is needed, which proves effective in making data loading balanced and close samples assigned to the same sub-site [16].

Next, the above-mentioned four stages of the proposed LDSDC are presented in the following subsections, respectively.

3.1 Local Clustering (Stage 1)

In general, the overall distribution of a dataset is too complex to model. LDSDC partitions a local dataset into several local clusters such that each local cluster has a relative simple distribution for easing the subsequent modeling procedure. Specifically, LDSDC divides $\mathcal{X}_w$ into local clusters by exploiting a density-based method [50], which has shown superior performance among established single-machine clustering methods. The procedure is detailed next.

First, a K-Nearest-Neighbors (KNNs) kernel is used to estimate the density of each sample. Given a sample $\mathbf{x}$, its KNNs kernel density, denoted by $\rho(\mathbf{x})$, is defined as

$$\rho(\mathbf{x}) = \beta \sum_{\mathbf{y} \in \mathcal{N}_K(\mathbf{x})} \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right\}, \tag{1}$$

where $\mathcal{N}_K(\mathbf{x})$ denotes the KNNs set of $\mathbf{x}$; $\sigma = \frac{1}{|\mathcal{X}_w|}\sum_{\mathbf{x}\in\mathcal{X}_w} \|\mathbf{x} - \mathbf{n}_K(\mathbf{x})\|$ is the mean of the distance between $\mathbf{x}$ and its Kth nearest neighbor $\mathbf{n}_K(\mathbf{x})$; β is a normalization factor. Intuitively, $\rho(\mathbf{x})$ measures how dense the sample distribution is around $\mathbf{x}$.

Second, LDSDC finds the samples with the locally maximal ρ value and marks them as core samples. Formally, the core sample set is defined as

$$\mathcal{O}_w = \left\{\mathbf{x} | \mathbf{x} \in \mathcal{X}_w, \rho(\mathbf{x}) \geq \max_{\mathbf{y}\in\mathcal{N}_K(\mathbf{x})} \rho(\mathbf{y})\right\}. \tag{2}$$

On the other hand, for a non-core sample $\mathbf{x}$, its higher-density nearest neighbor (HDNN) $\boldsymbol{\pi}(\mathbf{x})$ is defined as

$$\boldsymbol{\pi}(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{N}_K(\mathbf{x}), \rho(\mathbf{y})>\rho(\mathbf{x})}{\arg\min} \|\mathbf{x}-\mathbf{y}\|. \tag{3}$$

HDNN allows us to construct a directed graph $\mathcal{G}_w = (\mathcal{X}_w, \mathcal{E}_w)$, where each vertex is a sample and a direct edge exists from a non-core sample to its HDNN, namely, $\mathcal{E}_w = \{(\mathbf{x}, \boldsymbol{\pi}(\mathbf{x})) | \mathbf{x} \in \mathcal{X}_w \backslash \mathcal{O}_w\}$. Fig. 2 shows an illustrative example for $\mathcal{G}_w$, in which

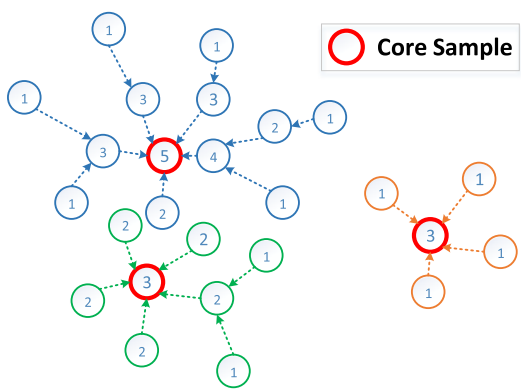


Fig. 2. Illustration for core samples and atom clusters. The atom clusters are represented by different colors and the core samples are marked using red bold circles. Trees rooted at the core samples form an acyclic graph. Each tree corresponds to an atom cluster.

each circle represents a sample with an integer indicating the magnitude of its kernel density for simplicity; a dotted arrow denotes a direct edge $(\mathbf{x}, \boldsymbol{\pi}(\mathbf{x}))$.

As we can see, in $\mathcal{G}_w$, starting from any non-core sample and following the directed edges, it will eventually reach a core sample. In other words, $\mathcal{G}_w$ can be partitioned into trees with disjoint vertices, where each tree is rooted at a core sample. A core sample and its descendants thus form a local cluster, called an *atom cluster*, which is denoted as $\mathcal{C}_s$.

3.2 Modelling Atom Clusters (Stage 2)

Atom clusters form the basis of final clusters, while a true cluster may consist of many atom clusters distributed in different sub-sites. Thus, a globally collecting step is needed to decide which atom clusters should be merged.

An atom cluster can be viewed as a local dense region of the given HD dataset, which is usually embedded in a LD subspace in the real world [14], [15]. In other words, the local samples though may be sparse in the whole HD space, usually are densely distributed in a LD subspace, in which the sample density is relatively stable. On the other hand, since the size of each atom cluster is usually small, to avoid overfitting [51], we do not pursue to exactly characterize the distribution of each atom cluster, but to give a "sketch" to it in an economical and efficient way. In this case, SGM proves competent in modeling HD samples embedded in LD subspace [48]. So we choose it as the sample model. Specifically, an atom cluster, denoted as $\mathcal{C}_s$, is modeled by a SGM characterized by a parameter set Φ_s. Then all the estimated parameter sets will be sent to the global site to further attain a global distribution density. How to efficiently estimate Φ_s for each $\mathcal{C}_s$ will be presented in Section 4.

3.3 Merging Models (Stage 3)

With the collected parameter sets of Φ_s, LDSDC defines a global density distribution by summing all local distributions. Specifically, for a sample $\mathbf{x}$ in the feature space, its global density $\rho_g(\mathbf{x})$ is given by

$$\rho_g(\mathbf{x}) = \sum_{s=1}^{S} \frac{|\mathcal{C}_s|}{|\mathcal{X}|} f(\mathbf{x}|\Phi_s), \tag{4}$$

where S is the total number of local models (atom clusters) and $f(\cdot|\Phi_s)$ denotes the probability density function of the SGM with parameter Φ_s. In practice, calculating $\rho_g(\mathbf{x})$ directly is time-consuming, so we use its log-lower-bound for the subsequent computations instead. Specifically, according to Jensen's inequality [52], the log-lower-bound $\underline{\rho_g}$ is given by

$$\log \rho_g(\mathbf{x}) \geq \sum_{s=1}^{S} \frac{|\mathcal{C}_s|}{|\mathcal{X}|} \log f(\mathbf{x}|\Phi_s) \triangleq \underline{\rho_g}(\mathbf{x}). \tag{5}$$

We empirically found that computing this bound can yield a factor of 1.5-3x speed-up with little accuracy loss.

Based on the log-lower-bound, the *connection value* between the ith and jth model $\alpha_{i,j}$ is defined as

$$\alpha_{i,j} = \min_{\theta \in [0,1]} \left\{ \underline{\rho_g}\big(\theta \boldsymbol{\mu}_i + (1-\theta)\boldsymbol{\mu}_j\big) \right\}, \tag{6}$$

where $\boldsymbol{\mu}_i \in \Phi_i$ and $\boldsymbol{\mu}_j \in \Phi_j$ denote the mean vector of the ith model and the jth model, respectively. Due to the non-convexity of (6), we approximate its solution via a simple but effective method as follows. First, divide the interval [0,1] into L consecutive intervals $[(l-1)/L, l/L], l = 1, \ldots, L$. Second, find a local minimal value for $\hat{\alpha}_{i,j}^l$ for $l = 1, \ldots, L$ by ternary search [53]. Finally, approximate the optimal solution to (6) by $\hat{\alpha}_{i,j} = \min_l \hat{\alpha}_{i,j}^l$. It can be shown that $\hat{\alpha}_{i,j} \to \alpha_{i,j}$ as $L \to \infty$.

Collecting connection values between different models can form a symmetric matrix $\mathbf{A} = \left[\hat{\alpha}_{i,j}\right]$ with size of S^2. The matrix $\mathbf{A}$ is normalized (with all its elements between 0 and 1) through the min-max normalization. Then, the models i and j are merged if $\hat{\alpha}_{i,j} \geq \delta$, where $\delta \in [0,1]$ is a threshold, thus yielding a merging table. Though δ can be set to any value in the range of $[0,1]$, there is only a small collection of δ values (its size is upper bounded by S) that affects the final merging. Furthermore, this collection together with δ can be found efficiently via solving a maximum spanning tree in the graph defined by the matrix $\mathbf{A}$ (by treating $\mathbf{A}$ as an adjacency matrix; see the FJDD algorithm in [50] for more details). In Section 5.5, we will discuss more about the choice of this parameter. Finally, LDSDC broadcasts this table to all the sub-sites.

3.4 Merging Local Clusters (Stage 4)

In this stage, each sub-site independently merges atom clusters according to the received merging table. Then the final clusters at each sub-site are generated accordingly. Let us conclude this section by a computational analysis to the proposed LDSDC algorithm.

Let N_w be the number of samples in the wth sub-site. The computational complexity of *Stage 1* is $O(N_w^2)$ since KNNs need to be computed for each sample. In *Stage 2*, the parameters of SGM can be estimated using Algorithm 1, whose complexity will be discussed in Section 4.2. In *Stage 3*, the cost of estimating (6) for a pair of models is $O(M^2C)$, where M is the data dimension and C is a small number (around 30 by our experience). We found that the connection value between two far apart models is usually negligibly small. Thus, we only consider $\sqrt{S}$ neighbors of each model in computing the connection value, where S is the number of models (atom clusters). As a result, the computational complexity of *Stage 3* is $O(S\sqrt{S}M^2C)$. Finally, *Stage 4* has a computational complexity of $O(N_w)$. The analysis of network transmission overhead is left in Section 4.1.

4 Subspace Gaussian Model

In this section, we present the fundamental SGM used for modeling each atom cluster (cf. Section 3.2). The formulation of SGM has been briefly presented by Bouveyron *et al.* [48] without in-depth discussion about the intuition behind it. In Section 4.1, we establish SGM in an illustrative manner to bridge the gap between the intuition and the formulation. Furthermore, we propose a novel fast parameter-estimation method for SGM, and then present a theoretical analysis of its correctness and convergency in Section 4.2.

4.1 Establishment of Subspace Gaussian Model

Although Gaussian model is a popular model in data representation, real HD samples usually distribute in a LD

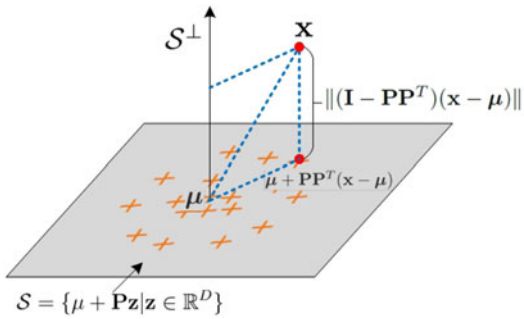


Fig. 3. A graphical interpretation of SGM.

subspace, and so the corresponding sample covariance matrix is nearly singular. This yields serious numerical problems in calculating the probability density of the standard Gaussian model. Furthermore, when the subspace dimension is far smaller than the original space dimension, the amount of parameters of the covariance matrix in the LD space may be significantly smaller than that in the HD space, implying higher overhead in computation and transmission cost for the latter. Some special Gaussian models have been proposed to reduce the number of parameters [33], but their assumption that sample components in different dimensions are statistically independent may not be very realistic.

Given an M-dimensional data space, SGM assumes that a singular Gaussian distribution is embedded in a D-dimensional affine subspace[1] $\mathcal{S}=\{\boldsymbol{\mu}+\mathbf{P}\mathbf{z}|\mathbf{z}\in\mathbb{R}^D\}$, where $\boldsymbol{\mu}\in\mathbb{R}^M$ and $\mathbf{P}\in\mathbb{R}^{M\times D}$ are respectively a shift vector and an orthonormal basis matrix of the D-dimensional subspace. Thus, the singular Gaussian distribution in this subspace is proportional to

$$\exp\Big\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{P}\mathbf{S}^{-1}\mathbf{P}^T(\mathbf{x}-\boldsymbol{\mu})\Big\}, \tag{7}$$

where $\mathbf{S}=\oplus_{d=1}^{D}\sigma_d$ is a diagonal matrix, i.e., the covariance matrix is diagonal under the basis matrix $\mathbf{P}$. For any sample $\mathbf{x}\in\mathbb{R}^M$, its density under SGM is related to two terms. As shown in Fig. 3, one is the density of $\mathrm{Prj}_{\mathcal{S}}\mathbf{x}$, where $\mathrm{Prj}_{\mathcal{S}}\mathbf{x}=\boldsymbol{\mu}+\mathbf{P}\mathbf{P}^T(\mathbf{x}-\boldsymbol{\mu})$ is the projection of $\mathbf{x}$ onto $\mathcal{S}$. The other is the distance from $\mathbf{x}$ to the affine subspace $\mathcal{S}$, which is $\|(\mathbf{I}-\mathbf{P}\mathbf{P}^T)(\mathbf{x}-\boldsymbol{\mu})\|$.

By incorporating the above two terms, the probability density function of SGM can be expressed as

$$\begin{aligned} f(\mathbf{x}|\boldsymbol{\mu},\mathbf{P},\mathbf{S},\sigma_0) =& \frac{1}{(2\pi)^{\frac{M}{2}}\sigma_0^{\frac{M-D}{2}}\sqrt{|\mathbf{S}|}} \\ &\times\exp\Big\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{P}\mathbf{S}^{-1}\mathbf{P}^T(\mathbf{x}-\boldsymbol{\mu})\Big\} \\ &\times\exp\Big\{-\frac{1}{2}\frac{\|(\mathbf{I}-\mathbf{P}\mathbf{P}^T)(\mathbf{x}-\boldsymbol{\mu})\|^2}{\sigma_0}\Big\}. \end{aligned} \tag{8}$$

On the right side of (8), the first term is the normalization factor; the second term models the singular Gaussian distribution in an affine subspace; the third term can be viewed as a decay factor which is exponentially related to the negative distance square from $\mathbf{x}$ to $\mathcal{S}$. From (8), one can see the total amount of the parameters ($\boldsymbol{\mu},\mathbf{P},\mathbf{S}$ and σ_0) is $M+M\times D+D+1\approx O(MD)$.

Compared with the standard Gaussian, SGM allows the covariance to be completely defined in an affine subspace. Furthermore, the total number of parameters of SGM is very flexible, which can easily adapt to the data dimension by choosing a suitable value of D. Note that when $D\ll M$, SGM has much smaller parameter size $O(MD)$ than the standard Gaussian with parameter size $O(M^2)$. Recalling that S is the total number of atom clusters and W is the number of subsites, from Fig. 1, it is evident that the network transmission overhead of our method is $S\times O(MD)+S\times W\approx O(S(MD+W))$. Furthermore, if the reshuffling pre-process is performed, the network transmission overhead will increase by $O(MN)$, where N is the sample set size.

4.2 Fast Parameter-Estimation for SGM

In this subsection, we derive a fast parameter-estimation method for SGM. Assume that $\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_D$ without loss of generality, and that $\sigma_D\geq\sigma_0$ so that the SGM can capture the principal structure of the distribution of data. Thus, given a sample set $\mathcal{X}=\{\mathbf{x}_n\in\mathbb{R}^M\}_{n=1}^N$, the maximum likelihood (ML) estimation of SGM can be formulated as the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\mu},\mathbf{P},\mathbf{S},\sigma_0}\quad & \sum_{n=1}^{N}\ln f(\mathbf{x}_n|\boldsymbol{\mu},\mathbf{P},\mathbf{S},\sigma_0) \\ \text{s.t.}\quad & \mathbf{P}^T\mathbf{P}=\mathbf{I}_D, \\ & \mathbf{S}=\bigoplus_{d=1}^{D}\sigma_d, \\ & \sigma_1\geq\sigma_2\geq\cdots\geq\sigma_D\geq\sigma_0>0. \end{aligned} \tag{9}$$

Solving problem (9) directly is difficult. Fortunately, by some transformations, (9) can be reformulated into a tractable form. To begin with, we introduce an auxiliary matrix $\mathbf{Q}\in\mathbb{R}^{M\times(M-D)}$ such that

$$[\mathbf{P},\mathbf{Q}]^T[\mathbf{P},\mathbf{Q}]=\mathbf{I}_M, \tag{10}$$

i.e., the columns of $\mathbf{Q}$ are a set of orthonormal basis vectors of the orthogonal complement of the range space $\mathcal{R}(\mathbf{P})$. Then, by letting

$$\boldsymbol{\Sigma}=[\mathbf{P},\mathbf{Q}](\mathbf{S}\oplus\sigma_0\mathbf{I}_{M-D})[\mathbf{P},\mathbf{Q}]^T, \tag{11}$$

(8) can be reformulated as

$$G(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})=\frac{1}{(2\pi)^{\frac{M}{2}}\sqrt{|\boldsymbol{\Sigma}|}}\exp\Big\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big\}, \tag{12}$$

which is nothing but the density function of a general Gaussian distribution in HD space [51]. Based on (12), the log-likelihood in (9) can be rewritten as

1. In this paper, we indiscriminately use the terms "affine subspace" and "subspace" for simplicity.

$$\begin{aligned}&\sum_{n=1}^{N}\ln G(\mathbf{x}_n|\boldsymbol{\mu},\boldsymbol{\Sigma})\\&=-\frac{1}{2}\sum_{n=1}^{N}\Big(M\ln 2\pi+\ln|\boldsymbol{\Sigma}|+(\mathbf{x}_n-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n-\boldsymbol{\mu})\Big)\\&=-\frac{N}{2}\Big(M\ln 2\pi+\ln|\boldsymbol{\Sigma}|+\mathrm{Tr}(\mathbf{C}\boldsymbol{\Sigma}^{-1})\Big),\end{aligned}\tag{13}$$

where

$$\mathbf{C}=\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})(\mathbf{x}_n-\boldsymbol{\mu})^T.\tag{14}$$

Notice that maximizing (13) is equivalent to minimizing $\ln|\boldsymbol{\Sigma}|+\mathrm{Tr}(\mathbf{C}\boldsymbol{\Sigma}^{-1})$, and thus the ML estimation of SGM is equivalently to solve

$$\begin{aligned}\min_{\boldsymbol{\mu},\mathbf{P},\mathbf{S},\sigma_0}\quad&\ln|\boldsymbol{\Sigma}|+\mathrm{Tr}(\mathbf{C}\boldsymbol{\Sigma}^{-1})\\\text{s.t.}\quad&\boldsymbol{\Sigma}=[\mathbf{P},\mathbf{Q}](\mathbf{S}\oplus\sigma_0\mathbf{I}_{M-D})[\mathbf{P},\mathbf{Q}]^T,\\&[\mathbf{P},\mathbf{Q}]^T[\mathbf{P},\mathbf{Q}]=\mathbf{I}_M,\\&\mathbf{S}=\bigoplus_{d=1}^{D}\sigma_d,\\&\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_D\geq\sigma_0>0.\end{aligned}\tag{15}$$

By setting the derivative of the objective function in (15) with respect to (w.r.t.) $\boldsymbol{\mu}$ to zero, the optimal solution for $\boldsymbol{\mu}$ is given by

$$\boldsymbol{\mu}^{\star}=\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.\tag{16}$$

Substituting this into (14), one can see that $\mathbf{C}$ is nothing but the sample covariance matrix. Then, to find optimal $\mathbf{P}$ and $\mathbf{Q}$, we need the following proposition.

Proposition 1. *Suppose that $\mathbf{C}=\mathbf{F}\boldsymbol{\Lambda}\mathbf{F}^T$ is the eigenvalue decomposition of $\mathbf{C}$ defined by (14). If the problem (15) is solvable, then there exists an optimal solution such that $[\mathbf{P}^{\star},\mathbf{Q}^{\star}]=\mathbf{F}$.*

Proof. Because $\mathbf{C}$ is symmetric and $\boldsymbol{\Sigma}$ is positive definite, by using the fact

$$\mathrm{Tr}\Big(\mathbf{C}\boldsymbol{\Sigma}^{-1}\Big)\geq\sum_{m=1}^{M}\lambda_m(\mathbf{C})\cdot\lambda_{M-m+1}\Big(\boldsymbol{\Sigma}^{-1}\Big)=\sum_{m=1}^{M}\frac{\lambda_m(\mathbf{C})}{\lambda_m(\boldsymbol{\Sigma})}.$$

(Theorem 4.3.53 in [54]), we have

$$\begin{aligned}\ln|\boldsymbol{\Sigma}|+\mathrm{Tr}\Big(\mathbf{C}\boldsymbol{\Sigma}^{-1}\Big)&=\sum_{m=1}^{M}\ln\lambda_m(\boldsymbol{\Sigma})+\mathrm{Tr}\Big(\mathbf{C}\boldsymbol{\Sigma}^{-1}\Big)\\&\geq\sum_{m=1}^{M}\ln\lambda_m(\boldsymbol{\Sigma})+\sum_{m=1}^{m}\frac{\lambda_m(\mathbf{C})}{\lambda_m(\boldsymbol{\Sigma})}.\end{aligned}\tag{17}$$

Suppose that $\boldsymbol{\Sigma}^{\star}$ is an optimal solution to problem (9). By using the result (17) and the eigenvalue decomposition of $\mathbf{C}=\mathbf{F}\boldsymbol{\Lambda}\mathbf{F}^T$, we have

$$\begin{aligned}\ln|\boldsymbol{\Sigma}^{\star}|+\mathrm{Tr}\Big(\mathbf{C}(\boldsymbol{\Sigma}^{\star})^{-1}\Big)&\geq\sum_{m=1}^{M}\ln\lambda_m(\boldsymbol{\Sigma}^{\star})+\sum_{m=1}^{M}\frac{\lambda_m(\mathbf{C})}{\lambda_m(\boldsymbol{\Sigma}^{\star})}\\&=\ln\Big|\mathbf{F}\Big(\bigoplus_{m=1}^{M}\lambda_m(\boldsymbol{\Sigma}^{\star})\Big)\mathbf{F}^T\Big|\\&\quad+\mathrm{Tr}\Big(\mathbf{C}\Big[\mathbf{F}\Big(\bigoplus_{m=1}^{M}\lambda_m(\boldsymbol{\Sigma}^{\star})\Big)\mathbf{F}^T\Big]^{-1}\Big).\end{aligned}\tag{18}$$

This implies that $\mathbf{F}\big(\oplus_{m=1}^{M}\lambda_m(\boldsymbol{\Sigma}^{\star})\big)\mathbf{F}^T$ is an optimal solution such that $[\mathbf{P}^{\star},\mathbf{Q}^{\star}]=\mathbf{F}$. □

It can be verified that problem (15) is solvable if $D<\mathrm{rank}(\mathbf{C})$. This condition can be easily satisfied as N is much larger than D in practice. Thanks to Proposition 1, let $[\mathbf{P},\mathbf{Q}]=\mathbf{F}$ and then solving (15) reduces to solving

$$\begin{aligned}\min_{\{\sigma_d\}_{d=0}^{D}}\quad&\sum_{d=1}^{D}\Big(\ln\sigma_d+\frac{\lambda_d(\mathbf{C})}{\sigma_d}\Big)+\sum_{m=D+1}^{M}\Big(\ln\sigma_0+\frac{\lambda_m(\mathbf{C})}{\sigma_0}\Big)\\\text{s.t.}\quad&\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_D\geq\sigma_0>0,\end{aligned}\tag{19}$$

and its optimal solution is given in the following proposition.

Proposition 2. *The optimal solution to (19) is given by*

$$\sigma_d^{\star}=\lambda_d(\mathbf{C})\quad(d=1,2,\ldots,D),\tag{20}$$

and

$$\sigma_0^{\star}=\frac{1}{M-D}\sum_{m=D+1}^{M}\lambda_m(\mathbf{C}).\tag{21}$$

So far we have derived the ML estimators of $\boldsymbol{\mu}$, $\mathbf{P}$, $\mathbf{S}$ and σ_0 (cf. (16), Propositions 1 and 2). However, calculating $\mathbf{P}^{\star}$ and $\mathbf{S}^{\star}$ needs the eigenvalue decomposition of the matrix $\mathbf{C}$, which leads to a computation complexity of $O(M^3)$. In addition, storing $\mathbf{C}$ requires a memory space of $O(M^2)$. Both the computation and the storage overheads are almost intolerable when M is large (e.g., $M>10^5$). To deal with this challenge, we propose a fast parameter-estimation method for the case $M^2\gg D\times N$.

First of all, due to

$$\begin{aligned}\sigma_0^{\star}&=\frac{1}{M-D}\sum_{m=D+1}^{M}\lambda_m(\mathbf{C})\\&=\frac{1}{M-D}\Big(\mathrm{Tr}(\mathbf{C})-\sum_{d=1}^{D}\lambda_d(\mathbf{C})\Big),\end{aligned}\tag{22}$$

only the D largest eigenvalues and the associated eigenvectors of $\mathbf{C}$ need to be computed. On the other hand, $\mathbf{C}$ can be expressed as $\mathbf{C}=\frac{1}{N}\bar{\mathbf{X}}\bar{\mathbf{X}}^T$, where $\bar{\mathbf{X}}=\mathbf{X}-\boldsymbol{\mu}\mathbf{1}_N^T\in\mathbb{R}^{M\times N}$ is the centralized sample matrix. Thus, the partial eigenvalue decomposition of $\mathbf{C}$ is equivalent to solving

$$\begin{aligned}\max_{\mathbf{P}}\quad&\frac{1}{2}\mathrm{Tr}(\mathbf{P}^T\bar{\mathbf{X}}\bar{\mathbf{X}}^T\mathbf{P})\\\text{s.t.}\quad&\mathbf{P}^T\mathbf{P}=\mathbf{I}_D.\end{aligned}\tag{23}$$

Inspired by the re-weighted optimization framework [55], we propose the following iterative two-step method to

solve problem (23)

$$1) \quad \text{Update } \mathbf{G} = \bar{\mathbf{X}}(\bar{\mathbf{X}}^T\hat{\mathbf{P}}); \tag{24a}$$

$$2) \quad \text{Solve } \hat{\mathbf{P}} = \underset{\mathbf{P}^T\mathbf{P}=\mathbf{I}_D}{\arg\max} \operatorname{Tr}(\mathbf{P}^T\mathbf{G}). \tag{24b}$$

Proposition 3. *Let $h(\mathbf{P})$ be the objective function of problem (23) and $\mathbf{P}_t$ denote the solution yielded by the two-step method (24a) and (24b) at the iteration t. Then $h(\mathbf{P}_t)$ increases with t monotonically and the sequence $\{\mathbf{P}_t\}$ converges to a KKT solution [52] to (23).*

Proof. Notice that $h(\mathbf{P})$ is strictly convex when $\mathbf{P} \in \mathcal{R}(\bar{\mathbf{X}})$ and $\nabla_{\mathbf{P}} h(\mathbf{P}) = \bar{\mathbf{X}}\bar{\mathbf{X}}^T\mathbf{P}$. If $\mathbf{P}_{t+1} \neq \mathbf{P}_t$, we have

$$h(\mathbf{P}_{t+1}) > h(\mathbf{P}_t) + \operatorname{Tr}\Big((\mathbf{P}_{t+1} - \mathbf{P}_t)^T \nabla_{\mathbf{P}} h(\mathbf{P}_t)\Big) \geq h(\mathbf{P}_t),$$

where the second inequality holds because

$$\mathbf{P}_{t+1} = \underset{\mathbf{P}^T\mathbf{P}=\mathbf{I}_D}{\arg\max} \operatorname{Tr}\big(\mathbf{P}^T \nabla_{\mathbf{P}} h(\mathbf{P}_t)\big). \tag{25}$$

Moreover, $h(\mathbf{P})$ is bounded above by $\frac{1}{2}\|\bar{\mathbf{X}}\|_F^2$. Therefore, $h(\mathbf{P}_t)$ increases with t monotonically and the sequence $\{\mathbf{P}_t\}$ must converge.

On the other hand, the Lagrangian of the optimization problem in (25) can be written as

$$\mathcal{L}(\mathbf{P}, \mathbf{A}) = \operatorname{Tr}\big(\mathbf{P}^T \nabla_{\mathbf{P}} h(\mathbf{P}_t)\big) + \operatorname{Tr}\big(\mathbf{A}^T\big(\mathbf{P}^T\mathbf{P} - \mathbf{I}_D\big)\big),$$

where $\mathbf{A} \in \mathbb{R}^{D\times D}$ is the Lagrange dual variable. Since $\mathbf{P}_{t+1}$ is the optimal solution to (25), according to the KKT condition, there exists an $\tilde{\mathbf{A}}$ such that

$$\nabla_{\mathbf{P}} \mathcal{L}(\mathbf{P}_{t+1}, \tilde{\mathbf{A}}) = \nabla_{\mathbf{P}} h(\mathbf{P}_t) + \mathbf{P}_{t+1}\big(\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^T\big) = \mathbf{0}. \tag{26}$$

When $\mathbf{P}_{t+1} = \mathbf{P}_t$, (26) becomes

$$\nabla_{\mathbf{P}} h(\mathbf{P}_{t+1}) + \mathbf{P}_{t+1}\big(\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^T\big) = \mathbf{0},$$

which implies $\mathbf{P}_{t+1}$ is also a KKT solution to (23). Thus we have completed the proof. □

The updating $\mathbf{G}$ in (24a) needs a computational complexity of $O(DMN)$. In (24b) for solving $\hat{\mathbf{P}}$, the optimal solution is known as $\hat{\mathbf{P}} = \mathbf{U}\mathbf{V}^T$ [56], where $\mathbf{U}$ and $\mathbf{V}$ are left and right singular vector matrices of $\mathbf{G} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ (singular value decomposition), respectively. Thus, (24b) requires a computational complexity of $O(D^2M)$. Finally, we come up with the fast ML estimation method in Algorithm 1, with the computational complexity of $O(DMN + D^2M)$. Furthermore, it only needs a memory space of $O(M(D+N))$.

Algorithm 1. Fast ML Estimation for SGM

Input: Sample matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$, Subspace dimension $D \in \mathbb{R}$
Output: $\hat{\boldsymbol{\mu}} \in \mathbb{R}^M, \hat{\mathbf{P}} \in \mathbb{R}^{M\times D}, \hat{\mathbf{S}} \in \mathbb{R}^{D\times D}, \hat{\sigma}_0$

$\hat{\boldsymbol{\mu}} = \frac{1}{N}\mathbf{X}\mathbf{1}_N$;
$\bar{\mathbf{X}} = \mathbf{X} - \hat{\boldsymbol{\mu}}\mathbf{1}_N^T$;
Initialize $\hat{\mathbf{P}}$ as $[\mathbf{I}_D, \mathbf{0}]^T \in \mathbb{R}^{M\times D}$;
repeat
  Update $\mathbf{G} = \bar{\mathbf{X}}(\bar{\mathbf{X}}^T\hat{\mathbf{P}})$;
  Decompose $\mathbf{G}$ into $\mathbf{U}\mathbf{D}\mathbf{V}^T$;
  Update $\hat{\mathbf{P}} = \mathbf{U}\mathbf{V}^T$;
until *Convergence*;
$\hat{\mathbf{S}} = \frac{1}{N}\mathbf{D}$;
$\hat{\sigma}_0 = \frac{1}{M-D}\Big(\frac{1}{N}\sum_{m=1}^{M} \bar{\mathbf{x}}_m^T\bar{\mathbf{x}}_m - \operatorname{Tr}(\hat{\mathbf{S}})\Big)$, where $\bar{\mathbf{x}}_m^T$ is the mth row of $\bar{\mathbf{X}}$;

5 Experiments

We evaluate the performance of the proposed LDSDC algorithm by experiment. All experiments are implemented on a server cluster comprising four machines. Each machine is equipped with an Intel E7-4809 v2 1.90G 48-core CPU, 60 GB memory, running Spark 2.1.0.

5.1 Baseline Methods and Parameter Setting

We select four state-of-the-art distributed clustering algorithms as baseline methods for performance comparison, including k-means‖ [57], DBDC [4], LSHDDP [32], REMOLD [16]. Note that REMOLD is a preliminary version of our work. Among the baseline methods, DBDC and LSHDDP share a common density meansure, which is described as follows.

- ϵ cut-off density. It is defined as the number of samples falling inside a ball of radius ϵ. The parameter ϵ is estimated as the 2 percent position of the ascending ordered distances among all sample pairs according to the recommendation in [32].

Table 1 lists detailed parameter settings for all the methods under test. The best performance is obtained for each method by searching throughout the parameter space stated in this table. To make the results stable and repeatable, the reshuffling pre-process is performed for all methods.

TABLE 1
Simulation Settings of All the Methods Under Test

Method	Setting
k-means‖	The parameter k is set to the real class number. We run it five times for each dataset with different random seeds.
DBDC	Use ϵ cut-off density. The parameter $MinPts$ is determined by $\lambda \cdot \bar{\rho}$, where λ is selected from $\{0.01, 0.05, 0.1, 0.5, 1\}$ and $\bar{\rho}$ is the mean density. Each outlier is assigned to its nearest cluster.
LSHDDP	Use ϵ cut-off density. The samples with top k gamma values (refer to [30]) are chosen as cluster centers, where k is the real class number. We set parameters $M = 10$ and $\pi = 3$ according to the recommendation in [32].
REMOLD	The parameter K is set to $\sqrt{N/W}$, where N and W are the numbers of samples and sub-sites, respectively. The parameter δ is enumerated in a candidate collection Δ, which is extracted by an auxiliary algorithm [16].
LDSDC (Proposed)	The parameter D is set to the smallest integer such that the retained eigenvalue ratio (i.e., the ratio of the sum of retained eigenvalues to the sum of total eigenvalues) of the sample covariance matrix (cf. (14)) is greater than 70%. All the other parameters are determined by the same way as REMOLD.

TABLE 2
Ten Synthetic Gaussian Datasets

Dataset	#Samples	#Dimensions	#Clusters	Number of Clusters with Embedded Dimension										#Partitions (sub-sites)
				2-D	4-D	8-D	16-D	32-D	64-D	128-D	256-D	512-D	1024-D	
G-1	50,000	2	20	20										4
G-2	100,000	4	40	20	20									4
G-3	150,000	8	60	20	20	20								4
G-4	200,000	16	80	20	20	20	20							4
G-5	250,000	32	100	20	20	20	20	20						8
G-6	300,000	64	120	20	20	20	20	20	20					8
G-7	350,000	128	140	20	20	20	20	20	20	20				16
G-8	400,000	256	160	20	20	20	20	20	20	20	20			16
G-9	450,000	512	180	20	20	20	20	20	20	20	20	20		32
G-10	500,000	1,024	200	20	20	20	20	20	20	20	20	20	20	32

The first column is dataset code, where G-l implies its dimension is 2^l. Columns from two to four are the number of samples, dimensions and clusters in each dataset, respectively. Columns from five to fourteen present the distribution of clusters with different embedding dimensions, where D-D is the abbreviation of D-dimension. The last column gives the number of partitions (i.e., sub-sites) for each dataset.

5.2 Datasets

We evaluate our method on three series of datasets, i.e., synthetic Gaussian datasets, synthetic norm-ball surface datasets and real-world datasets, which are presented in the following subsections, respectively.

5.2.1 Synthetic Gaussian Datasets

First, we test all the five algorithms on the most common Gaussian datasets. Each dataset consists of Gaussian clusters with various embedding dimensions. Specifically, an M-dimensional cluster with D embedding dimension is generated through the following procedure.

1) Generate a Gaussian random (element-wise) matrix $\mathbf{T} \in \mathbb{R}^{M\times D}$ and normalize it by $\frac{\mathbf{T}}{\|\mathbf{T}\|_F}$.
2) Generate a shift vector $\boldsymbol{\mu} \in \mathbb{R}^M$ whose elements are independently sampled from a uniform distribution over the interval $[-\frac{20}{\log M}, \frac{20}{\log M}]$.
3) A sample is generated as

$$\mathbf{x} = \mathbf{T}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{27}$$

where $\mathbf{z} \sim G(\mathbf{0}, \mathbf{I}_D)$ and $\boldsymbol{\epsilon} \sim G(\mathbf{0}, \frac{1}{10M}\mathbf{I}_M)$.
4) Repeat 3) 2,500 times, and these samples constitute a cluster (i.e., each cluster consists of 2,500 samples).

TABLE 3
Ten Synthetic Norm-Ball Surface Datasets

Dataset	#Samples	#Dimensions	#Clusters	Surface Set	#Partitions
S-1	1500	2	3	$\mathbb{S}(2)$	2
S-2	3,000	4	3	$\mathbb{S}(3)$	2
S-3	6,000	8	3	$\mathbb{S}(4)$	2
S-4	12,000	16	3	$\mathbb{S}(5)$	4
S-5	24,000	32	3	$\mathbb{S}(6)$	4
S-6	48,000	64	3	$\mathbb{S}(7)$	4
S-7	96,000	128	3	$\mathbb{S}(8)$	8
S-8	182,000	256	3	$\mathbb{S}(9)$	8
S-9	364,000	512	3	$\mathbb{S}(10)$	16
S-10	728,000	1,024	3	$\mathbb{S}(11)$	32

The first column is dataset code, where S-l implies its dimension is 2^l. Columns from two to four are the numbers of samples, dimensions and clusters in each dataset, respectively. The fifth column shows the corresponding norm-ball surface set $\mathbb{S}(D) = \{0.5\mathcal{E}_{0.5}^{D,2^{D-1}}, \mathcal{E}_1^{D,2^{D-1}}, 1.5\mathcal{E}_2^{D,2^{D-1}}\}$. The last column gives the number of partitions (i.e., sub-sites) for each dataset.

Ten datasets with dimension from 2 to 1,024 are generated and detailed information about these datasets are listed in Table 2.

5.2.2 Synthetic Norm-Ball Surface Datasets

Unlike Gaussian datasets, in which clusters are shaped like ellipsoids and not very difficult in clustering, synthetic norm-ball surface datasets include clusters with non-convex shapes, which is more challenging to most methods. Specifically, an ℓ_p-norm ball surface in D-dimensional space refers to the set $\mathcal{E}_p^D = \{\mathbf{z}|\mathbf{z} \in \mathbb{R}^D, \|\mathbf{z}\|_p = 1\}$. Furthermore, by left multiplying a semi-orthonormal matrix $\mathbf{P} \in \mathbb{R}^{M\times D}(D < M)$, $\mathcal{E}_p^D$ can be embedded in an M-dimensional space. We denote the embedded norm-ball surface as $\mathcal{E}_p^{D,M} = \{\mathbf{P}\mathbf{z}|\mathbf{z} \in \mathbb{R}^D, \|\mathbf{z}\|_p = 1\}$. Thus norm-ball surface datasets are generated by sampling from three different norm-ball surfaces $0.5\mathcal{E}_{0.5}^{D,M}$, $\mathcal{E}_1^{D,M}$ and $1.5\mathcal{E}_2^{D,M}$, where D and M are chosen to produce data with different dimensions. We generate ten datasets whose detailed information is given in Table 3, and Fig. 4 gives two illustrative examples.

5.2.3 Real-World Datasets

We select five HD real-world datasets for our experiments. Among them, Salinas, PaviaC and PaviaU are three hyperspectral images (HSIs) [58]. Salinas was collected by the AVIRIS sensor over agricultural crops in Salinas Valley, California,

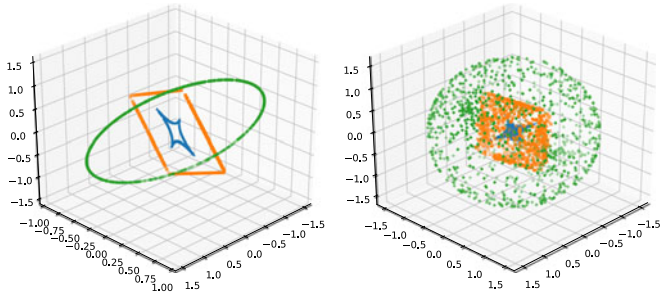


Fig. 4. Two illustrative examples for three norm-ball surface datasets, including (a) an example sampled from $0.5\mathcal{E}_{0.5}^{2,3}$ (blue), $\mathcal{E}_1^{2,3}$ (orange) and $1.5\mathcal{E}_2^{2,3}$ (green), and (b) an example sampled from $0.5\mathcal{E}_{0.5}^{3,3}$ (blue), $\mathcal{E}_1^{3,3}$ (orange) and $1.5\mathcal{E}_2^{3,3}$ (green).

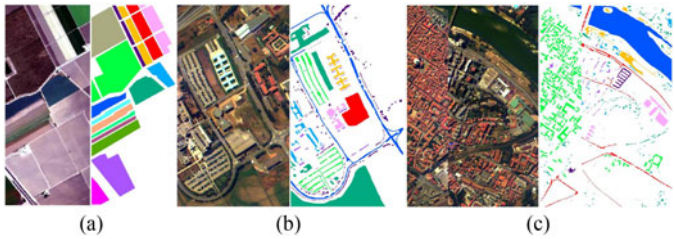


Fig. 5. Hyperspectral datasets of color composite images together with the ground truth maps for (a) Salinas, (b) PaviaC, and (c) PaviaU, respectively.

and is characterized by high spatial resolution (3.7-meter pixels). Its size is 512 lines by 217 samples, in addition to 204 spectral bands used. It has 16 classes including vegetables, bare soils, and vineyard fields. PaviaU and PaviaC are two scenes acquired by the ROSIS sensor during a flight over Pavia, nothern Italy. The spatial size is 610×340 for PaviaU and 1096×715 for PaviaC. PaviaU and PaviaC were acquired using 103 spectral bands and 102 spectral bands, respectively. Both of them contain 9 classes. Fig. 5 shows the color composite images and the ground truth maps for the three datasets. We also consider another real-world dataset MNIST [59], containing 60,000 examples of handwritten digits from 0 to 9, and each example has a dimension of 784 (28×28). Fig. 6 shows some samples of MNIST. MNIST1M [60] is an extended version of MNIST with 1,000,000 samples. Table 4 lists the detailed information about these datasets.

5.3 Performance Metric

We evaluate the clustering quality of all the algorithms under test by two measures: normalized mutual information (NMI) and purity. The larger the two measures, the better the clustering quality.

NMI [61] is a clustering metric from the information-theory, and it quantifies the "amount of information" as follows. Given a dataset V of size N, suppose there are I clusters and J true classes. Let N_i, $N^{(j)}$ and $N_i^{(j)}$ denote the number of samples in cluster i, true class j, and both cluster i and true class j, respectively. Then, NMI is computed by

$$\mathrm{NMI} = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{N_i^{(j)}}{N}\log\frac{NN_i^{(j)}}{N_i N^{(j)}}}{\sum_{i=1}^{I}\frac{N_i}{N}\log\frac{N_i}{N}\sum_{j=1}^{J}\frac{N^{(j)}}{N}\log\frac{N^{(j)}}{N}}.$$

Purity [62] counts the number of correctly assigned samples divided by the size of the dataset. Given a dataset with size N, suppose that $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_I\}$ and $\{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_J\}$ are the set of clusters and true classes, respectively. The purity is defined by

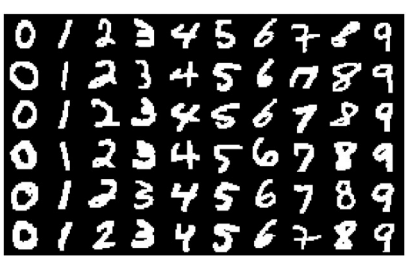


Fig. 6. Some typical samples of the MNIST dataset.

TABLE 4
Detailed Information for Five Real-World Datasets

Dataset	#Samples	#Dimensions	#Clusters	#Partitions
Salinas	111,104	206	16	8
PaviaU	207,400	105	9	8
PaviaC	783,640	104	9	16
MNIST	60,000	784	10	4
MNIST1M	1,000,000	784	10	32

Columns from two to four are the number of samples, dimensions and clusters in each dataset, respectively. The last column gives the number of partitions (i.e., sub-sites) for each dataset.

$$\mathrm{purity} = \frac{1}{N}\sum_{i=1}^{I}\max_{1\le j\le J}|\mathcal{C}_i \cap \mathcal{W}_j|.$$

Notice that *purity* cannot be used for trade-off between the quality of the clustering versus the number of clusters (e.g., dividing a dataset into singletons can get the perfect score), so we report the purity only for the clustering result with the highest value of NMI.

In addition, we also report the amount of transmission cost and the running time (machine time spent on the whole clustering process) for each method in the conducted experiments.

5.4 Experimental Results

5.4.1 Results on Synthetic Gaussian Datasets

The clustering results for the Gaussian datasets are shown in Fig. 7.

From Fig. 7a, one can observe that, in terms of NMI measure, LSHDDP (denoted as "+") performs best; both LDSDC (denoted as "△") and REMOLD (denoted as "✱") also perform well with comparable performance; k-means||

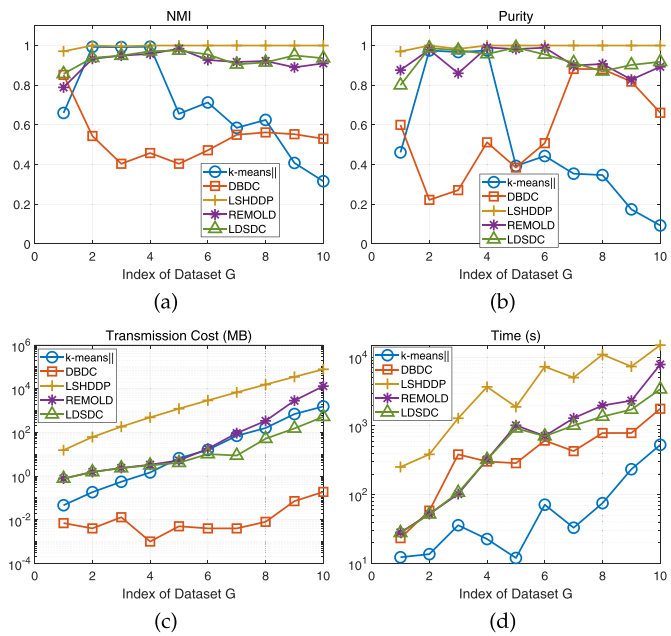


Fig. 7. Performance comparison of the five methods under test on the Gaussian datasets. (a) NMI, (b) purity, (c) transmission cost (mega bytes in semi-log scale) (d) running time (seconds in semi-log scale), w.r.t. the index of the datasets, respectively.

(denoted as "○") achieves almost perfect performance for the LD datasets G-2, G-3 and G-4, but its performance seriously degrades as data dimension increases possibly due to many local minima in HD datasets; and DBDC (denoted as "□") performs worst except for the LD dataset G-1 because the model used is too simplified to accurately model the complex distribution in HD datasets. In terms of the purity measure, the corresponding performances of all the algorithms are shown in Fig. 7b. The performance behaviors of k-means||, LSHDDP, REMOLD and LDSDC observed from Fig. 7a also apply to Fig. 7b. DBDC shows different low-quality performance behaviors in Figs. 7a and 7b, although its performance is comparable to LDSDC and REMOLD for the datasets of G-7, G-8 and G-9 in Fig. 7b, indicating that it may over-divide the datasets into finer clusters.

The corresponding transmission cost (mega bytes) is presented in Fig. 7c. A common performance behavior for all the algorithms except DBDC can be seen from this figure, showing that the transmission cost linearly increases with the data dimension in the semi-log scale. Furthermore, LSHDDP has the heaviest transmission cost due to reshuffling the dataset several times; k-means|| has lower transmission costs (proportional to the product of the size and the dimension of the dataset) for LD datasets G-1 through G-4 than LDSDC and REMOLD, while LDSDC has lower transmission costs for HD datasets G-6 through G-10 than k-means|| and REMOLD. It is noticeable that their transmission costs are almost the same for the dataset G-5 (median dimension), and specifically, REMOLD needs to transmit 13 gigabytes but the cost of LDSDC is just 0.5 gigabytes for the HD dataset G-10. DBDC has the lowest transmission cost since the amount of the parameters for transmission in its own operation rule is not sensitive to the volume of datasets.

The running time (seconds) comparison for all the methods under test is shown in Fig. 7d. Again, a common performance behavior for all the algorithms can be seen from this figure, showing that the trend of running time linearly increases with the data dimension in semi-log scale. The trends associated with LSHDDP, LDSDC and REMOLD have similar growth rate, but LSHDDP (with highest NMI score) has much higher running time than LDSDC and REMOLD due to extraordinary repeated distance calculations of numerous sample pairs. Instead, LDSDC (with smaller growth rate than REMOLD for G-7 through G-10) achieves NMI and purity performances comparable to LSHDDP at the expense of more economical computation cost. For instance, the former can handle the dataset G-10 within an hour, but the latter needs over four hours. The running time for DBDC is basically smaller than that for both LDSDC and REMOLD together with smaller growth rate. Finally, the running time for k-means|| is the lowest, showing its high efficiency in handling large but LD datasets.

5.4.2 Results on Synthetic Norm-Ball Surface Datasets

The clustering results for the norm-ball surface datasets are shown in Fig. 8.

Fig. 8a exhibits the NMI scores for all the methods under test. Some observations from this figure are as follows. The proposed LDSDC (△) significantly outperforms all the other

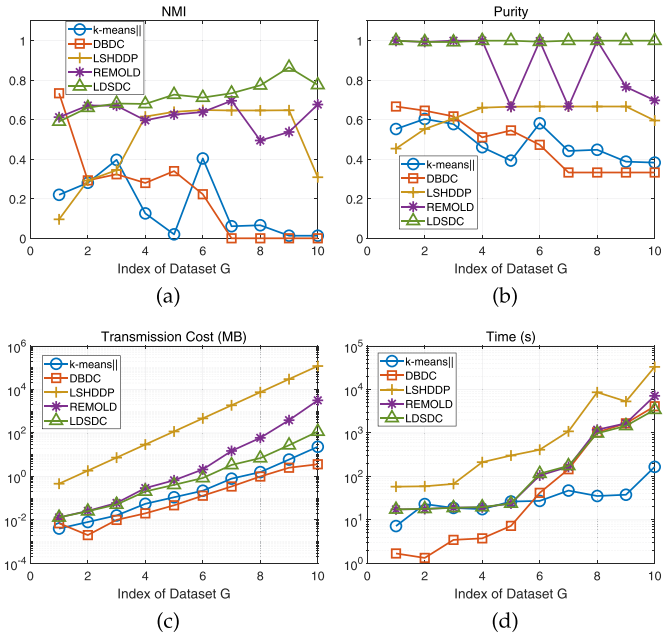


Fig. 8. Performance comparison of the five methods under test on the norm-ball surface datasets. (a) NMI, (b) purity, (c) transmission cost (mega bytes in semi-log scale), (d) running time (seconds in semi-log scale), w.r.t. the index of the datasets, respectively.

methods with NMI scores between 0.6 and 0.85, and its performance is insensitive to both of the data dimension and cluster structure. Overall, REMOLD (✱) performs second and LSHDDP (+) performs third. However, k-means|| (○, designed for classifying convex clusters) and DBDC (□, with insufficient core samples in modeling the HD distribution) basically fail for all the datasets despite the highest NMI score of the latter for the LD dataset S-1. In terms of purity, as shown in Fig. 8b, the performance ranking and behaviors of all the algorithms are basically consistent with those associated with Fig. 8a, namely, the proposed LDSDC performs best with almost perfect purity scores (near 1), REMOLD performs second with purity scores between 0.65 and 1, and LSHDDP performs third with purity scores below 0.65, and k-means|| and DBDC share the bottom performance. These experimental results demonstrate that the proposed LDSDC algorithm is much robust to complex non-convex clusters and data dimension than all the other algorithms.

The corresponding transmission cost (mega bytes in semi-log scale) is presented in Fig. 8c. The linear-increase behavior of transmission cost in semi-log scale observed from Fig. 7c can also be observed from Fig. 8c for all the algorithms under test. Though their growth rates are also similar, their transmission cost ranks are LSHDDP (rank 1, highest), REMOLD (rank 2), LDSDC (rank 3), k-means|| (rank 4) and DBDC (rank 5, lowest). It is noticeable that REMOLD and LDSDC seem to share similar network costs for the first four datasets, but the former expends the cost for the HD dataset S-10 about 30 times higher than the latter due to larger growth rate in semi-log scale. As for the running time comparison (seconds in semi-log scale) shown in Fig. 8d, basically, the linear-increase trends (in semi-log scale) versus the data dimension associated with Fig. 7d also apply to Fig. 8d for all the algorithms under test, showing that their running time ranks are

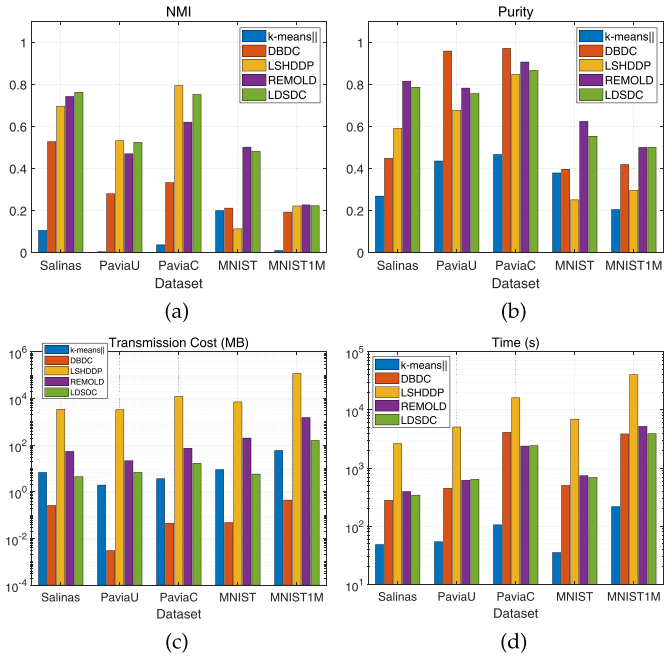


Fig. 9. Performance comparison of the five methods under test on the five real-world datasets by bar graph, including (a) NMI, (b) purity, (c) transmission cost (mega bytes in semi-log scale) and (d) running time (seconds in semi-log scale).

LSHDDP (rank 1, highest), REMOLD (rank 2), LDSDC (rank 3), k-means|| (rank 5 over HD datasets) and DBDC (rank 5 over LD datasets). Finally, let us mention that LDSDC and REMOLD expend almost the same running time over all the datasets except for the HD dataset S10, and that k-means|| has a stable low running-time cost for all the datasets (owing to its simple computation at each iteration), while DBDC has an amazingly low running-time cost for the first four datasets but its running-time cost rapidly increases for the other HD datasets. Overall, the above experimental results on various synthetic datasets have demonstrated that the proposed LDSDC outperforms the other four sate-of-the-art algorithms.

5.4.3 Results on Real-World Datasets

Fig. 9 shows the performance comparison of the five algorithms using five real-world datasets. One can see from Fig. 9a that in terms of NMI, LDSDC (green bar) gets the hightest score only for Sainas, and the second highest score for the other four datasets. However, its NMI performance is much worse for the two datasets MNIST and MNST1M possibly due to the binary nature of the data (either around 0 for black pixels or around 255 for white pixels), thereby causing lower modeling accuracy since the SGM itself is a continuous distribution model. However, we would like to emphasize that the performance of LDSDC is either the best, or the second best but very comparable to the best for each dataset, thereby exhibiting its performance consistence (and robustness) over the five datasets. Among the five methods, only REMOLD (denoted by purple bar) achieves competitive performance compared to LDSDC, while the other algorithms cannot maintain consistent good performance over the five datasets.

Fig. 9b shows the corresponding performance comparison results in terms of the purity score. From this figure, one can observe that the performance of REMOLD is either the best, or the second best but comparable to the best for each dataset, thereby exhibiting its performance consistence (and robustness) over the five datasets. Nevertheless, the performance of the proposed LDSDC is very comparable to that of REMOLD over all the datasets. Hence, these results also support LDSDC's good performance quality and consistence over the five datasets.

The corresponding transmission cost (mega bytes) and running time (seconds) are shown in Figs. 9c and 9d (semi-log plots), respectively. As mentioned above, the performance qualities (NMI and purity) of k-means|| (denoted by blue bar), DBDC (denoted by red bar) and LSHDDP (denoted by dark yellow bar) are neither satisfactory nor consistent over the five datasets. Instead, let us only focus on transmission cost and running time associated with REMOLD and LDSDC. One can observe, from Figs. 9c and 9d, that their running times are comparable to each other, while the transmission cost associated with latter is much lower than the former (in semi-log scale). Therefore, these experimental results on the five real-world datasets well support the overall superior efficacy (performance quality, computation cost and network overhead) of the proposed LDSDC algorithm over the other four state-of-the-art algorithms.

5.5 Parameter Sensitivity Analysis

The clustering performance of the proposed LDSDC depends on the number of sub-sites (NoS) and three user-specified parameters, i.e., K (number of K NNs), D (dimension of subspace in SGM) and δ (threshold for model merging). We investigate the performance sensitivity of these parameters with seven representative datasets (G-6, G-7, S-7, S-8, Salinas, PaviaU and MNIST) in the following subsections, respectively.

5.5.1 Sensitivity to Number of Sub-Sites

In this experiment, LDSDC is tested under the default parameter settings given in Table 1, and the experimental results are obtained for NoS between 2 and 18 such that the running time is within a reasonable range for each dataset for ease of illustration. From Fig. 10, one can observe that the NMI scores decrease slowly with NoS in trend, but not very sensitive to NoS in spite of some fluctuations for S-7 and S-8 datasets. As for the metric of purity, similar performance behavior can be observed in Fig. 10b. On the other hand, as shown in Fig. 10c, the transmission costs increase linearly in trend, though the linear trends for G-7 and MNIST are much higher than for the other datasets due to their diverse cluster distributions and high-dimensionality. Finally, as shown in Fig. 10d, the running times decrease with NoS exponentially instead.

In conclusion, the increase of the NoS will lead to slight degradation of the clustering performance and moderate or mild growth in the transmission cost on one hand, exponentially higher efficiency on the other hand.

5.5.2 Sensitivity to K and D

Let $\mathrm{RER} \triangleq \frac{\sum_{i=1}^{D} \lambda_i(\mathbf{C})}{\mathrm{Tr}(\mathbf{C})}$ be the retained eigenvalue ratio of the data covariance matrix $\mathbf{C}$ (cf. (14)) in the D-dimensional

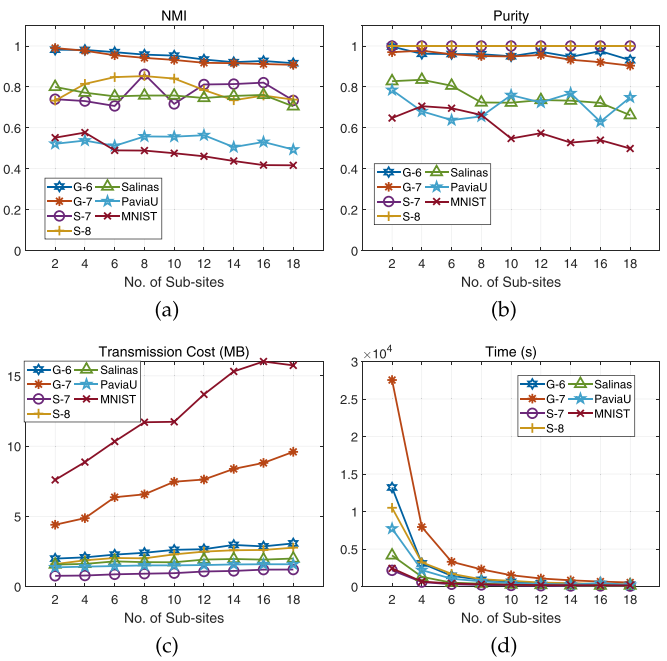


Fig. 10. The clustering performance of LDSDC as a function of the number of sub-sites, including (a) NMI, (b) purity, (c) transmission cost (mega bytes) and (d) running time (seconds).

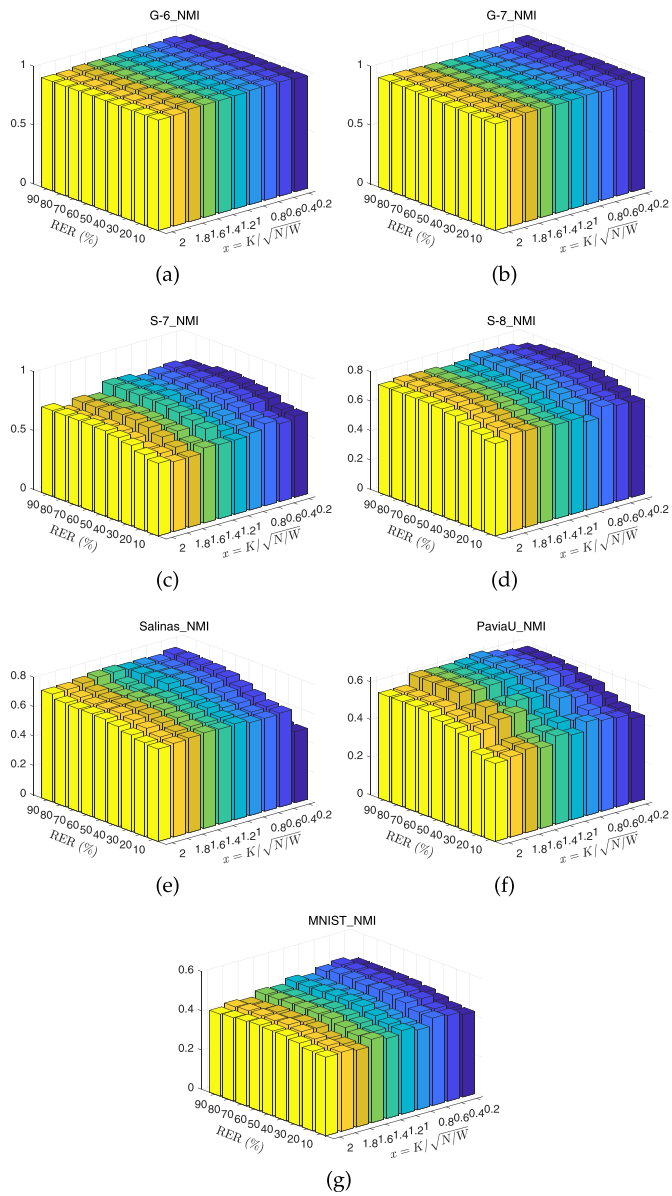


Fig. 11. The clustering performance (measured by NMI) of LDSDC as a function of $x = K/\sqrt{N/W}$ and RER % (which is larger for larger D) for datasets (a) G-6, (b) G-7, (c) S-7, (d) S-8, (e) PaviaU, (f) Salinas and (g) MNIST, respectively.

subspace of SGM. Obviously, the larger the RER, the larger the parameter D. The experimental results for NMI versus $x = K/\sqrt{N/W}$ and RER (instead of D) are displayed in Fig. 11 where $0.2 \leq x \leq 2$ and $10\% \leq \text{RER} \leq 90\%$ for each dataset. Overall, the variation of NMI is mild, indicating LDSDC's low performance sensitivity to (K, RER); the smaller the value of K or the larger the value of RER, the better the NMI performance. Moreover, as shown in Fig. 11a through Fig. 11d, the NMI performance is better for G-6 and G-7 (synthetic Gaussian datasets) than for S-7 and S-8 (synthetic norm-ball surface datasets) besides larger fluctuations for the latter; as shown in Fig. 11e through Fig. 11g, it is better for Salinas and PaviaU datasets than for MNIST dataset in spite of slightly larger fluctuations for PaviaU.

In summary, the performance of LDSDC is not very sensitive to K and RER, for which a good choice is $K \approx \sqrt{N/W}$ and $\text{RER} \geq 70\%$.

5.5.3 Sensitivity to δ

In this subsection, for the parameter $\delta \in [0, 1]$, we present the qualitative analysis, followed by the quantitative analysis. First of all, the larger δ, the larger the number of clusters, implying that a positive correlation between them. As mentioned in Section 3.3, only a small collection of finite δ values, denoted as the set Δ here, can affect the final clustering. Moreover, the mapping from $\delta \in \Delta$ to the number of clusters is bijective (one-to-one and onto). Hence, LDSDC can uniquely yield the clusters with the desired clustering number. However, if the prior information about the clustering number is not available, LDSDC can produce a series of hierarchical results by enumerating each $\delta \in \Delta$, thereby providing a comprehensive perspective to the underlying structure in the dataset.

To further analyze how δ affects the clustering performance, the NMI performances of the LDSDC are displayed in Fig. 12 for all the datasets in our experiments as δ varies. One can see, from the left plot (Gaussian datasets) in Fig. 12, that the NMI starting from $\delta = 0$ increases as δ increases either mildly or rapidly like mountain climbing, and finally reaches the top of the mountain (a plateau or roughly a plateau) for $\delta \geq 0.8$. From the mid plot (norm-ball surface datasets) in Fig. 12, we can see that each NMI increases with δ in a very nonlinear manner due to complex distributions of the datasets, and the maximum NMI appears for $\delta \geq 0.8$ except the dataset S-9 for which NMI has been on a plateau for $\delta \in [0.6, 0.77]$. As for the right plot (real-world datasets) in Fig. 12, each NMI increases with δ in a less tractable (more mild) manner than that associated with Gaussian datasets (norm-ball surface datasets), and it reaches either a plateau or the maximum over [0.8,1].

In conclusion, in practical applications, we suggest that users can choose a suitable δ from the intersection

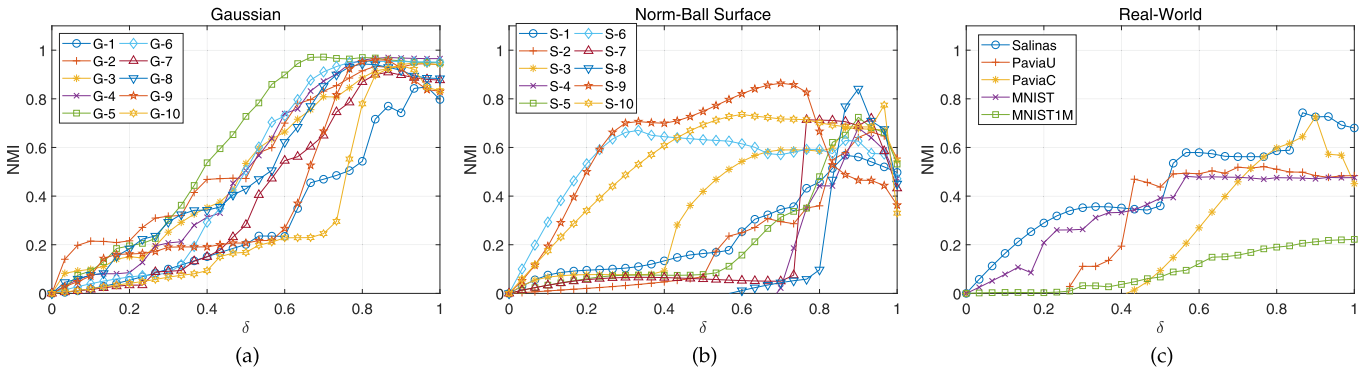


Fig. 12. The clustering performance (measured by NMI) of LDSDC as a function of δ for (a) Gaussian datasets, (b) norm-ball surface datasets, and (c) real-world datasets.

of [0.8,1] and Δ through the elbow method based on intrinsic metrics [1].

6 Conclusion

Based on the fact that local samples of HD data usually reside in LD data spaces, we have presented a distributed clustering algorithm (LDSDC algorithm) as shown in Fig. 1, especially for HD datasets with complex cluster structures. The proposed LDSDC algorithm uses the SGM for modeling every local cluster (atom cluster) in the LD space (conducted in every sub-site), and then get upgraded SGMs for all the collected local atom clusters in the global site together with a cluster merging table (determined by the threshold parameter δ for model merging) for each sub-site to yield the final clustering result. Some experimental results using synthetic datasets and real-world datasets were presented to demonstrate the effectiveness of the proposed LDSDC algorithm and its overall superior performance (clustering quality, network overhead, and computation cost) over four benchmark algorithms (k-means||, DBDC, LSHDDP and REMOLD), in spite of some performance degradation when the given datasets are basically constituted by finite discrete feature values (e.g., datasets MNIST and MNIST1M). Finally an analysis for the proposed LDSDC about its performance sensitivity to the system parameters was presented.

Some further studies are left in the future, including extension of the proposed LDSDC from facilitator sub-site networks to P2P networks, design of an automatic strategy for the parameter δ, and development of a privacy-preserving LDSDC due to increasing concerns of personal privacy.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2017YFC1501503), Beijing Natural Science Foundation (No. L191016), National Social Science Foundation of China (No. 18CSH019), Beijing Municipal Education Commission Research Program (No. SM20191001107, PXM2019_014213_000007), China Scholarship Council Program (No. 201907090062), and Ministry of Science and Technology, ROC (No. 109-2221-E-007-024).

References

[1] J. Han *et al.*, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann., 2011.

[2] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 103, no. 23, pp. 8577–8582, 2006.

[3] C. D. Wang, J.-H. Lai, and J.-Y. Zhu, "Graph-based multiprototype competitive learning and its applications," *IEEE Trans. Syst., Man Cybern. C*, vol. 42, no. 6, pp. 934–946, Nov. 2012.

[4] E. Januzaj *et al.*, "DBDC: Density based distributed clustering," in *Proc. Int. Conf. Extending Database Technol.*, 2004, pp. 88–105.

[5] W. Gan *et al.*, "Data mining in distributed environment: A survey," *Wiley Interdisciplinary Rev.: Data Mining Knowl. Discov.*, vol. 7, no. 6, 2017, Art. no. e1216.

[6] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*. Berlin, Germany: Springer, 2006, pp. 25–71.

[7] A. Rosato *et al.*, "Recent advances on distributed unsupervised learning," in *Proc. Int. Workshop Neural Netw.*, 2015, pp. 77–86.

[8] N. Kushwaha and M. Pant, "Link based BPSO for feature selection in big data text clustering," *Future Gener. Comput. Syst.*, vol. 82, pp. 190–199, 2018.

[9] Y. Gu *et al.*, "Clustering-driven unsupervised deep hashing for image retrieval," *Neurocomputing*, vol. 368, pp. 114–123, 2019.

[10] J. M. Murphy and M. Maggioni, "Unsupervised clustering and active learning of hyperspectral images with nonlinear diffusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1829–1845, Mar. 2019.

[11] A. Fahad *et al.*, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 267–279, Sep. 2014.

[12] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[13] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Comput. Statist. Data Anal.*, vol. 71, pp. 52–78, 2014.

[14] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[15] K. Chakrabarti and S. Mehrotra, "Local dimensionality reduction: A new approach to indexing high dimensional spaces," in *Proc. Int. Conf. Very Large Data Bases*, 2000, pp. 89–100.

[16] M. Liang, Q. Li, Y.-A. Geng, J. Wang, and Z. Wei, "REMOLD: An efficient model-based clustering algorithm for large datasets with spark," in *Proc. IEEE 23rd Int. Conf. Parallel Distrib. Syst.*, 2017, pp. 376–383.

[17] Q. Tong *et al.*, "Efficient distributed clustering using boundary information," *Neurocomputing*, vol. 275, pp. 2355–2366, 2018.

[18] L. Zeng *et al.*, "Distributed data mining: A survey," *Inf. Technol. Manage.*, vol. 13, no. 4, pp. 403–409, 2012.

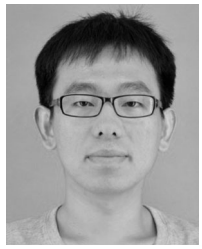
[19] K. M. Hammouda and M. S. Kamel, "Models of distributed data clustering in peer-to-peer environments," *Knowl. Inf. Syst.*, vol. 38, no. 2, pp. 303–329, 2014.

[20] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A K-means clustering algorithm," *J. Roy. Statist. Soc.. Series C (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 1979.

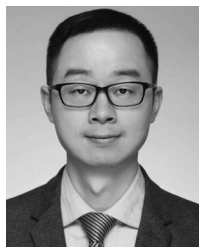
[21] J. Vaidya and C. Clifton, "Privacy-preserving K-means clustering over vertically partitioned data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 206–215.

[22] G. Ji and X. Ling, "Ensemble learning based distributed clustering," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2007, pp. 312–321.

[23] N. K. Visalakshi and K. Thangavel, "Distributed data clustering: A comparative analysis," in *Foundations of Computational, Intelligence Volume 6. Studies in Computational Intelligence*, vol. 206. Berlin, Germany: Springer, 2009, pp. 371–397.
[24] J. Jeong *et al.*, "Integration of distributed biological data using modified k-means algorithm," in *Proc. Int. Conf. Emerg. Technol. Knowl. Discov. Data Mining*, 2007, pp. 469–475.
[25] M.-F. F. Balcan *et al.*, "Distributed k-means and k-median clustering on general topologies," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1995–2003.
[26] X. Xu *et al.*, "A fast parallel clustering algorithm for large spatial databases," *Data Mining Knowl. Discov.*, vol. 3, no. 3, pp. 263–290, 1999.
[27] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, vol. 96, pp. 226–231.
[28] E. Januzaj *et al.*, "Scalable density-based distributed clustering," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, 2004, pp. 231–244.
[29] N. I. Wei-Wei *et al.*, "Local density based distributed clustering algorithm," *J. Softw.*, vol. 19, no. 9, pp. 2339–2348, 2008.
[30] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
[31] S. Gong and Y. Zhang, "EDDPC: An efficient distributed density peaks clustering algorithm," *J. Comput. Res. Develop.*, vol. 53, pp. 1400–1409, 2016.
[32] Y. Zhang, S. Chen, and G. Yu, "Efficient distributed density peaks for clustering large data sets in MapReduce," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3218–3230, Dec. 2016.
[33] H.-P. Kriegel, P. Kroger, A. Pryakhin, and M. Schubert, "Effective and efficient distributed model-based clustering," in *Proc. 5th IEEE Int. Conf. Data Mining*, 2005, pp. 258–265.
[34] A. P. Dempster *et al.*, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. Series B (Methodol.)*, vol. 39, pp. 1–38, 1977.
[35] X. Lin *et al.*, "Privacy-preserving clustering with distributed EM mixture modeling," *Knowl. Inf. Syst.*, vol. 8, no. 1, pp. 68–81, 2005.
[36] I. T. Jolliffe, *Principal Component Analysis and Factor Analysis*. New York, NY, USA: Springer, 2002, pp. 150–166.
[37] S. Mika *et al.*, "Kernel PCA and de-noising in feature spaces," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1999, pp. 536–542.
[38] T. Hofmann *et al.*, "Kernel methods in machine learning," *The Ann. Statist.*, vol. 36, pp. 1171–1220, 2008.
[39] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Roy. Statist. Soc. Series B (Methodol.)*, vol. 61, no. 3, pp. 611–622, 1999.
[40] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, Dec. 2005.
[41] E. J. Candès *et al.*, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 11:1–11:37, 2011.
[42] R. Agrawal *et al.*, "Automatic subspace clustering of high dimensional data," *Data Mining Knowl. Discov.*, vol. 11, no. 1, pp. 5–33, 2005.
[43] I. Assent, "Clustering high dimensional data," *Wiley Interdisciplinary Rev., Data Mining Knowl. Discov.*, vol. 2, no. 4, pp. 340–350, 2012.
[44] C.-H. Cheng *et al.*, "Entropy-based subspace clustering for mining numerical data," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 1999, pp. 84–93.
[45] H. Nagesh *et al.*, "Adaptive grids for clustering massive data sets," in *Proc. SIAM Int. Conf. Data Mining*, 2001, pp. 1–17.
[46] C. M. Procopiuc *et al.*, "A Monte Carlo algorithm for fast projective clustering," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2002, pp. 418–427.
[47] K. Kailing *et al.*, "Density-connected subspace clustering for high-dimensional data," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 246–256.
[48] C. Bouveyron *et al.*, "High-dimensional data clustering," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 502–519, 2007.
[49] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annu. Symp. Comput. Geometry*, 2004, pp. 253–262.
[50] Y.-A. Geng *et al.*, "RECOME: A new density-based clustering algorithm using relative KNN kernel density," *Inf. Sci.*, vol. 436-437, pp. 13–30, 2018.
[51] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[52] C.-Y. Chi *et al.*, *Convex Optimization for Signal Processing and Communications: From Fundamentals to Applications*. Boca Raton, FL, USA: CRC Press, 2017.
[53] M. S. Bajwa, A. P. Agarwal, and S. Manchanda, "Ternary search algorithm: Improvement of binary search," in *Proc. IEEE Int. Conf. Comput. Sustainable Global Develop.*, 2015, pp. 1723–1725.
[54] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge Univ. Press, 2012.
[55] F. Nie *et al.*, "Optimal mean robust principal component analysis," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, vol. 32, pp. 1062–1070.
[56] J. Huang *et al.*, "Spectral rotation versus k-means in spectral clustering," in *Proc. 27th AAAI Conf. Artif. Intell.*, 2013, pp. 431–437.
[57] B. Bahmani *et al.*, "Scalable k-means++," *Proc. VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.
[58] Hyperspectral remote sensing scenes, Accessed: Feb. 2020. [Online]. Available: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes
[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[60] G. Loosli *et al.*, "Training invariant support vector machines using selective sampling," in *Large Scale Kernel Machines*. Cambridge, MA, USA: MIT Press, 2007, pp. 301–320.
[61] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 583–617, 2002.
[62] R. R. Larson, "Introduction to information retrieval," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 61, no. 4, pp. 852–853, 2010.

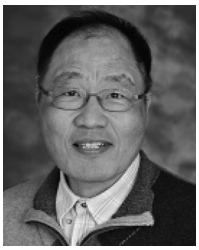


Yangli-ao Geng received the BS degree in information management and information system from Zhengzhou University, Zhengzhou, China. He is currently working toward the PhD degree in the School of Computer Science and Technology, Beijing Jiaotong University, Beijing, China. His research interests include data mining and machine learning.



Qingyong Li (Member, IEEE) received the BS degree in computer science and technology from Wuhan University, Wuhan, China, in 2001, and the PhD degree in computer science and technology from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2006. He is currently a professor with Beijing Jiaotong University, Beijing. His research interests include computer vision and artificial intelligence.

Mingfei Liang received the master's degree in computer science from Beijing Jiaotong University, Beijing, China, in 2018. He is currently an algorithmic engineer with Tencent, Beijing, China. His research interests include data mining and big data.



Chong-Yung Chi (Fellow, IEEE) received the BS degree from the Tatung Institute of Technology, Taipei, Taiwan, in 1975, the master's degree from National Taiwan University, Taipei, Taiwan, in 1977, and the PhD degree from the University of Southern California, Los Angeles, California, in 1983 all in electrical engineering. Currently, he is a professor of National Tsing Hua University, Hsinchu, Taiwan. He has published more than 240 technical papers, including more than 85 journal papers (mostly in the *IEEE Trans. Signal Processing*), more than 140 peer-reviewed conference papers, three book chapters, and two books, including a recent textbook, Convex Optimization for Signal Processing and Communications from Fundamentals to Applications, CRC Press, 2017 (which has been popularly used in a series of invited intensive short courses at the top-ranking universities in Mainland China since 2010 before its publication). Recently, he received 2018 IEEE Signal Processing Society Best Paper Award. His current research interests include signal processing for wireless communications, convex analysis and optimization for blind source separation, biomedical and hyperspectral image analysis, and graph signal processing. He has been a Technical Program Committee member for many IEEE sponsored and co-sponsored workshops, symposiums and conferences on signal processing and wireless communications, including co-organizer and general co-chairman of 2001 IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC). He was an associate editor (AE) for four IEEE Journals, including the *IEEE Trans. Signal Processing* for nine years (5/2001–4/2006, 1/2012–12/2015), and he was a member of Signal Processing Theory and Methods Technical Committee (SPTM-TC) (2005–2010), a member of Signal Processing for Communications and Networking Technical Committee (SPCOM-TC) (2011–2016), and a member of Sensor Array and Multichannel Technical Committee (SAM-TC) (2013–2018), IEEE Signal Processing Society.

Juan Tan received the PhD degree in management from Hunan University, Changsha, China. She is currently an associate professor with the Beijing Technology and Business University, Beijing, China. Her research interests include data mining and innovation management.

Heng Huang received the BS and MS degrees from Shanghai Jiao Tong University, Shanghai, China, in 1997 and 2001, respectively, and the PhD degree in computer science from Dartmouth College, Hanover, New Hampshire, in 2006. He is a John A. Jurenko Endowed professor in computer engineering with the Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, Pennsylvania. He is also a consulting researcher with JD Finance American Corporation. His research interests include machine learning, big data mining, bioinformatics, and neuroinformatics.