Computational Visual Media
https://doi.org/10.1007/s41095-022-0293-5 Vol. 9, No. x, month 2022, xx–xx

Research Article

PMSSC: Parallelizable Multi-Subset Based Self-Expressive Model for Subspace Clustering

Katsuya Hotta[1], Takuya Akashi[2], Shogo Tokai[1], and Chao Zhang[1](✉)

Abstract Subspace clustering methods which embrace a self-expressive model that represents each data point as a linear combination of other data points in the dataset provide powerful unsupervised learning techniques. However, when dealing with large datasets, representation of each data point by referring to all data points via a dictionary suffers from high computational complexity. To alleviate this issue, we introduce a parallelizable multi-subset based self-expressive model (PMS) which represents each data point by combining multiple subsets, with each consisting of only a small proportion of the samples. The adoption of PMS in subspace clustering (PMSSC) leads to computational advantages because the optimization problems decomposed over each subset are small, and can be solved efficiently in parallel. Furthermore, PMSSC is able to combine multiple self-expressive coefficient vectors obtained from subsets, which contributes to an improvement in self-expressiveness. Extensive experiments on synthetic and real-world datasets show the efficiency and effectiveness of our approach in comparison to other methods.

Keywords Subspace Clustering, Self-expressive model, Big data, Subsetting

1 Introduction

In many real-world cases, approximating high-dimensional data as a union of low-dimensional subspaces is a beneficial technique for reducing computational complexity and the effects of noise. The task of subspace clustering [1, 2], which is the segmentation of a set of data points into those lying on certain subspaces, has been studied in many practical applications such as face clustering [3], image segmentation [4], motion segmentation [5], scene segmentation [6], and homography detection [7]. Recently, self-expressive models [8, 9] have been explored, which embrace the self-expressive property of subspaces to compute an affinity matrix. The self-expressive property states that each data point from a union of subspaces can be represented as a linear combination of other points. Specifically, given a data matrix $X \in \mathbb{R}^{D\times N}$ in which each data point is a column, the self-expressive model of data point $\boldsymbol{x}_i \in \mathbb{R}^D$ can be described as

$$\boldsymbol{x}_i = X\boldsymbol{c}_i, \quad c_{ii} = 0, \tag{1}$$

where $\boldsymbol{c}_i \in \mathbb{R}^N$ is a coefficient vector, and the constraint $c_{ii} = 0$ avoids the trivial solution of representing a point as a linear combination of itself. The feasible solutions of Eq. (1) are generally not unique because the number of data points lying on a subspace is larger than its dimensionality. However, at least one $\boldsymbol{c}_i$ exists where c_{ij} is nonzero only if data points $\boldsymbol{x}_i, \boldsymbol{x}_j$ are in the same subspace, and such a state is called subspace-preserving [10]. Previous works have tried to compute subspace-preserving representations by imposing a regularization term on the coefficients $\boldsymbol{c}_i$. In particular, one algorithm for obtaining a sparse solution to Eq. (1), sparse subspace clustering (SSC) [8, 9], can recover subspaces under mild conditions by regularizing the coefficient matrix $C := [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N] \in \mathbb{R}^{N\times N}$ corresponding to the coefficient vector of each data point $\boldsymbol{x}_i$. SSC not only achieves high clustering accuracy for datasets with outliers and missing entries, but also has the useful properties of giving theoretical guarantees and providing modeling flexibility, which have influenced many other approaches such as [11, 12]. However, SSC suffers from high computational and memory costs when dealing with a large-scale dataset because of the need to determine the $\mathcal{O}(N^2)$ coefficients of C. In light of these problems, there has been much interest in recent years in

1 University of Fukui, 910-8507, Japan. E-mail: k-hotta@u-fukui.ac.jp, tokai@u-fukui.ac.jp, zhang@u-fukui.ac.jp (✉).

2 Iwate University, Morioka, 020-8550, Japan. E-mail: akashi@iwate-u.ac.jp.

Manuscript received: 2022-01-01; accepted: 2022-01-01

developing scalable subspace clustering algorithms that can be applied to large-scale datasets, taking advantage of the ease of analyzing computational complexity due to the simplicity of the model.

Several works have attempted to address the problem of computational cost for large-scale datasets using a sampling strategy, motivated by the sparsity assumption that each data point can be represented as a linear combination of a few basis vectors. The self-expressive property with a few sampled data points and classifying of the other data points was proposed in [13]. While this strategy can produce clustering results more efficiently for a large-scale dataset than directly applying SSC to all data, it leads to poor clustering performance when the sampled data is not representative of the original dataset. Although a learning-based sampling method has also been proposed for generating a coefficient matrix that is representative of the original dataset [14], the accuracy and computational complexity still depend largely on the size of the subsets, as these methods attempt to solve for a self-expressive model in a single subset. Also, no effort has been made to explicitly improve the self-expressiveness of the self-expressive coefficient vectors in these methods.

To further improve self-expressiveness without increasing the computational burden, in this paper, we propose a self-expressive model adopting multiple subsets, which is computable in parallel. Specifically, our model obtains a self-expressive coefficient matrix by combining multiple subsets; each subset consists of only a small proportion of the samples. This strategy not only enjoys the benefit of low computational cost like other single subset-based methods, but also is more effectively subspace-preserving because the representation of the original data is a linear combination of multiple self-expressive coefficient vectors.

Our contributions are highlighted as follows:

- a novel clustering approach that exploits a self-expressive model based on multiple subsets,
- a concisely formulated model,
- each subset can be computed independently in parallel without additional computational overhead,
- extensive experiments on both synthetic data and real-world datasets showing that our proposed method can achieve better results without increasing processing time.

2 Related Work

2.1 Background

In the past few years, there has been a surge of spectral clustering-based algorithms that segment a set of data points by performing spectral clustering. Previous classical methods, such as k-subspaces [15] and median k-flats [16], assume that the dimensionalities of the underlying subspaces are given in advance. This latent knowledge is generally hard to access in many real-world applications. In addition, these methods are usually non-convex and thus sensitive to initialization [17, 18]. Aiming to relax the limitations of the k-subspace algorithm, the majority of modern subspace clustering methods explored have turned to spectral clustering [19, 20], which segment data using an affinity matrix that captures whether a certain pair of data points lie on the same subspace. While many early methods [12, 21–23] achieve better segmentation than classical clustering algorithms even without the latent knowledge, these methods produce erroneous segmentation results for data points near the intersection of two subspaces due to the dense sampling of points lying on the subspace [24]. We now introduce previous subspace clustering approaches based on spectral clustering, then describe various techniques of scalable subspace clustering methods for dealing with large-scale datasets, which are closer to our proposed method.

2.2 Subspace Clustering Using Spectral Clustering

Most subspace clustering approaches based on spectral clustering consist of two phases: (i) computing an affinity matrix based on the nonzero coefficients that appear in the representation of each data point as a combination of other points, and (ii) segmenting data points from the computed affinity matrix by applying spectral clustering. The key to the success of segmentation is the phase of computing the affinity matrix. Therefore, many methods have been proposed to compute the affinity matrix. For example, local subspace affinity [24] and spectral curvature clustering (SCC) [25] find neighborhoods based on the observation that a point and its k-nearest neighbors often lie on the same subspace. However, the computational complexity of finding multi-way similarity in these methods grows exponentially with the number of subspace dimensions, motivating the use of a sampling strategy to lower the computational complexity [9]. Recently, the self-expressive model, which employs the self-expressive property in Eq. (1), has become the most popular one. In particular, SSC takes advantage of sparsity [26] by adopting ℓ_1 norm regularization of the coefficient vector to achieve high clustering performance. This idea has motivated many methods, using the ℓ_2 norm in least squares regression [27], the nuclear norm in low rank representation (LRR) [28], the ℓ_1 plus ℓ_2 norm in elastic net subspace clustering (EnSC) [29], and the Frobenius norm in efficient dense subspace clustering [30]. In practice, however, solving the

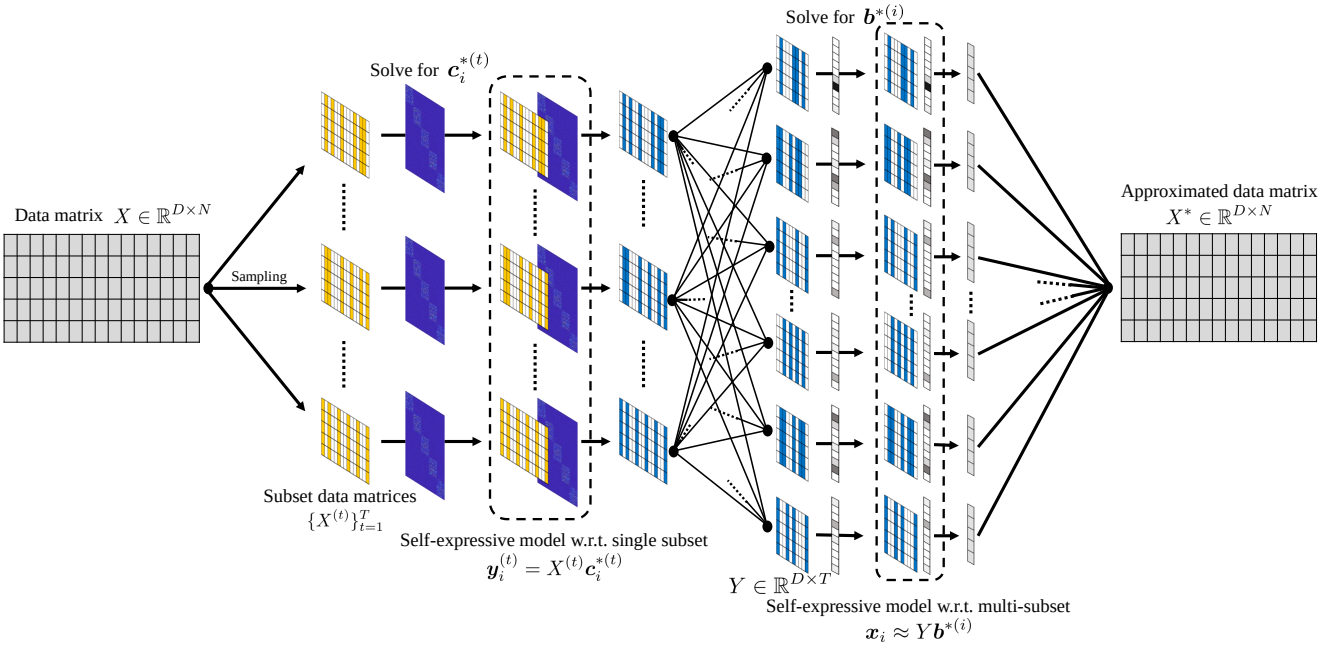


Fig. 1 Overview of our self-expressive model. Given a data matrix X in which each data point x_i is a column, our approach represents a self-expressive model over the entire data by combining multiple subsets generated by sampling (Algorithm 1). Specifically, our method computes the self-expressive data point $\boldsymbol{y}_i^{(t)}$ by solving for the self-expressive coefficient vector $\boldsymbol{c}_i^{*(t)}$ for each point $\boldsymbol{x}_i$ in T subsets (Algorithm 2). Then, the self-expressive properties of the entire data are obtained by solving for $\boldsymbol{b}^*$ using Y with each data point $\boldsymbol{y}_i^{(t)}$ computed from each subset as columns (Algorithm 3).

ℓ_p norm minimization problem for large-scale data may be prohibitive. Also, the memory required becomes larger as the amount of data increases.

2.3 Scalable Subspace Clustering

When constructing the affinity matrix, several methods based on spectral clustering suffer from high computational complexity. To reduce the computational complexity of this phase, a sparse self-expressive model adopting a greedy algorithm was proposed in [10, 31]. However, these approaches lead to unsatisfactory clustering results if the nonzero elements do not contain sufficient connections within each optimized coefficient vector [32]. Other popular approaches to alleviate the computational and memory loads were inspired by a sampling strategy. In [13], scalable sparse subspace clustering (SSSC) is computationally efficient, using a subset generated by random sampling. However, because the random sampling method relies on a single subset, data points from the same subspace will not be represented by the self-expressive model if they are not appropriately sampled. Exemplar-based subspace clustering [12, 33] is an efficient sampling technique that iteratively selects the least well-represented point as a subset to address the problem. Selective sampling-based scalable sparse subspace clustering (S^{5}C) [14], which generates a subset by selective sampling, provides approximation guarantees of the subspace-preserving property. In [34], the subspace-preserving representations are found by solving a consensus problem with multiple subsets to improve the connectivity of the affinity matrix. In [35], a divided-and-conquer framework using multiple subsets obtained by separating the entire dataset is proposed. While this approach can deal with large-scale data, final segmentation results depend on the self-expressive properties of the optimized self-expressive coefficient vectors of each subset. Our method differs significantly from [35] and [34] in that our proposed self-representation model is designed to minimize the difference from the original data points by combining the self-expressive property of multiple subsets. Lastly, in this paper, we limit our discussion to non-deep learning approaches, which are more mathematically straightforward to explain and rely less on parameter tuning.

3 Parallelizable Multi-Subset Based Sparse Subspace Clustering

3.1 Problem and Approach

As a problem definition, our final goal is to find the self-expressive coefficient vector $\boldsymbol{c}_i$, which satisfies the subspace-preserving representation in Eq. (1). That is, the

self-expressive residual can be obtained by solving the following optimization problem,

$$\min_{\boldsymbol{c}_i} \|\boldsymbol{x}_i - X\boldsymbol{c}_i\|_2^2 \quad \text{such that} \quad \|\boldsymbol{c}_i\|_0 \leqslant s,\ c_{ii} = 0, \quad (2)$$

where $\|\cdot\|_0$ is the ℓ_0 pseudo-norm that returns the number of nonzero entries in the vector. This optimization problem has been shown [36, 37] to recover provably subspace-preserving solutions using the orthogonal matching pursuit (OMP) algorithm [38]. s is a tuning parameter for the OMP algorithm, which controls the sparsity of the solution by selecting up to s entries in the coefficient vector $\boldsymbol{c}_i$. Although the OMP algorithm is computationally efficient and is guaranteed to give subspace-preserving solutions under mild conditions, it is unable to produce a subspace-preserving solution with a number of nonzero entries exceeding the dimensionality of the subspace [10]. This leads to poor clustering performance with too sparse affinity between data points, especially when the density of data points lying on the subspace is low.

We propose a novel subspace clustering algorithm with a parallelizable multi-subset based self-expressive model, as illustrated in Fig. 1. Sec. 3.2 introduces our proposed self-expressive model that extends the model in Eq. (2) to multiple subsets via a sampling technique. Sec. 3.3 then explains the solution of our self-expressive model by the OMP algorithm. Finally, we summarize the proposed subspace clustering algorithm in Algorithm 4.

3.2 Parallelizable Multi-Subset based Self-Expressive Model

To deal with large-scale data, we first generate T index subsets from the whole dataset by weighted random sampling [39] as follows:

$$\mathcal{I}^{(t)} \subset [N] \quad \text{s.t.} \quad (\mathcal{I}^{(t)}) = \lceil \delta N \rceil,\ t = 1, \ldots, T, \quad (3)$$

where $\mathcal{I}^{(t)}$ is the index set of the t-th subset that is sampled with probability proportional to the elements of the weight vector $\boldsymbol{w}^{(t)} \in \mathbb{R}^N$, $[N]$ is N indices $\{1, \ldots, N\}$, $0 < \delta \leqslant 1$ is the sampling rate, and $n(\cdot)$ is the cardinality function that is a measure of the number of elements. The t-th selected element of $\boldsymbol{w}^{(t)}$ is updated as $\boldsymbol{w}_i^{(t+1)} = 0.1\boldsymbol{w}_i^{(t)}$. Then, in each sampled t-th subset, the optimization problem in Eq. (2) can be expressed as follows:

$$\begin{aligned} \boldsymbol{c}_i^{*(t)} = &\arg\min_{\boldsymbol{c}_i^{(t)}} \|\boldsymbol{x}_i^{(t)} - X^{(t)}\boldsymbol{c}_i^{(t)}\|_2^2 \\ &\text{s.t. } \|\boldsymbol{c}_i^{(t)}\|_0 \leqslant s, \quad c_{ii}^{(t)} = 0, \end{aligned} \quad (4)$$

where $X^{(t)} \in \mathbb{R}^{D \times N}$ is the data matrix of the randomly sampled t-th subset. $\boldsymbol{c}_i^{(t)} \in \mathbb{R}^N$ is the self-expressive coefficient vector for each data point $\boldsymbol{x}_i^{(t)}$ in the t-th subset. Note that to ensure the dimensionality of $\boldsymbol{c}_i^{(t)}$ is N, the columns of each data matrix $X^{(t)}$ corresponding to the non-sampled indices are replaced by zero-vectors: $\boldsymbol{x}_i^{(t)} = \boldsymbol{0},\ \forall i \notin \mathcal{I}^{(t)}$. From each optimized coefficient vector $\boldsymbol{c}_i^{*(t)}$, each data point $\boldsymbol{x}_i^{(t)}$ can be represented by a self-expressive model, given by:

$$\boldsymbol{y}_i^{(t)} = X^{(t)}\boldsymbol{c}_i^{*(t)} \quad \text{s.t.} \quad c_{ii}^{*(t)} = 0, \quad (5)$$

where $\boldsymbol{y}_i^{(t)}$ is the data point computed by the self-expressive model from the t-th subset. In practice, however, the data point $\boldsymbol{y}_i^{(t)}$ in Eq. (5) generally has an error term $\boldsymbol{z}_i$, i.e., $\boldsymbol{y}_i^{(t)} = \boldsymbol{x}_i + \boldsymbol{z}_i$, because of the limitations of using $X^{(t)}$ as a dictionary for reconstruction. To minimize $\boldsymbol{z}_i$, we first represent $\boldsymbol{x}_i$ as a linear combination of $\boldsymbol{y}_i^{(t)}$, as follows:

$$\begin{aligned} \boldsymbol{x}_i &\approx \sum_{t=1}^{T} b_t^{(i)} (X^{(t)}\boldsymbol{c}_i^{*(t)}) \\ &\approx \sum_{t=1}^{T} b_t^{(i)} \boldsymbol{y}_i^{(t)}, \end{aligned} \quad (6)$$

where $\boldsymbol{b}^{(i)} \in \mathbb{R}^T$ is the weight coefficient vector to represent $\boldsymbol{x}_i$, and $b_t^{(i)} \in \mathbb{R}$ is the t-th entry of $\boldsymbol{b}^{(i)}$. The coefficient vector $\boldsymbol{b}^{(i)}$ of the linear combination in Eq. (6) can be obtained by solving the following optimization problem,

$$\boldsymbol{b}^{*(i)} = \arg\min_{\boldsymbol{b}^{(i)}} \|\boldsymbol{x}_i - \sum_{t=1}^{T} b_t^{(i)} \boldsymbol{y}_i^{(t)}\|_2^2. \quad (7)$$

For simplicity, we introduce a data matrix $Y = [\boldsymbol{y}_i^{(1)}, \ldots, \boldsymbol{y}_i^{(T)}] \in \mathbb{R}^{D \times T}$ with each data point $\boldsymbol{y}_i^{(t)}$ from Eq. (5) as columns, and rewrite Eq. (7) as

$$\boldsymbol{b}^{*(i)} = \arg\min_{\boldsymbol{b}^{(i)}} \|\boldsymbol{x}_i - Y\boldsymbol{b}^{(i)}\|_2^2. \quad (8)$$

This is the formulation of the optimization problem for subspace clustering in Eq. (2), and can be further described as:

$$\boldsymbol{x}_i \approx Y\boldsymbol{b}^{*(i)}. \quad (9)$$

Unlike in Eq. (1), here Y is the data matrix computed from each subset to represent $\boldsymbol{x}_i$. Thus, no constraint is required to avoid the trivial solution of representing a point as a linear combination of itself. To explicitly express Eq. (2), the self-expressive coefficient vector $\boldsymbol{c}_i^*$ corresponding to X is obtained by

$$\boldsymbol{c}_i^* = \sum_{t=1}^{T} b_t^{*(i)} \boldsymbol{c}_i^{*(t)}. \quad (10)$$

It is worth noting that each $\boldsymbol{c}^{*(t)}$ can be determined independently from each subset, so can be computed in parallel for speed.

Algorithm 1 Optimization for the parallelizable multi-subset based self-expressive model (PMS)

Input: Data matrix $X \in \mathbb{R}^{D\times N}$, number of subsets T, sampling rate δ, maximum number of repetitions s, error term ϵ

Generate T index subsets $\{\mathcal{I}^{(t)}\}_{t=1}^{T}$ via Eq. (3);
Generate T subset data matrices $\{X^{(t)}\}_{t=1}^{T}$ based on $\{\mathcal{I}^{(t)}\}_{t=1}^{T}$;
for $i = 1, \ldots, N$ **do**
  for $t = 1, \ldots, T$ **do**
    Given $X^{(t)}$ and $\boldsymbol{x}_i^{(t)}$, solve for $\boldsymbol{c}_i^{*(t)}$ via Algorithm 2;
    Given $\boldsymbol{c}_i^{*(t)}$, compute $\boldsymbol{y}_i^{(t)}$ for all data points via Eq. (5);
  end for
  Set $Y = [\boldsymbol{y}_i^{(1)}, \ldots, \boldsymbol{y}_i^{(T)}] \in \mathbb{R}^{D\times T}$;
  Given Y and $\boldsymbol{x}_i$, solve for $\boldsymbol{b}^{*(i)}$ via Algorithm 3;
  Given $\boldsymbol{c}_i^{*(t)}$ and $\boldsymbol{b}^{*(i)}$, compute $\boldsymbol{c}_i^{*}$ via Eq. (10);
end for
Set $C^* = [\boldsymbol{c}_1^*, \ldots, \boldsymbol{c}_N^*] \in \mathbb{R}^{N\times N}$;

Output: Coefficient matrix C^*

3.3 Optimization with Orthogonal Matching Pursuit

In this section, we show that the parameters of the proposed PMS model can be determined by dividing the optimization problem into two small optimization problems as summarized in Algorithm 1. Overall, Eq. (10) can be determined by solving the minimization problems in Eqs. (4) and (8). We introduce both of the minimization procedures below. Specifically, initially, T subset data matrices $\{X^{(t)}\}_{t=1}^{T}$ are generated based on the sampling set $\mathcal{I}^{(t)}$ in Eq. (3).

To efficiently solve for $\boldsymbol{c}_i^{*(t)}$ in Eq. (4), we introduce Algorithm 2 based on the OMP algorithm. The support set and the residual are initialized to $\mathcal{S} = \emptyset$ and $\boldsymbol{r} = \boldsymbol{x}_i^{(t)}$, respectively. $\mathcal{S}$ denotes the index set, which is updated on each iteration by adding one index j^*. j^* is computed using

$$j^* = \underset{j\in\mathcal{I}^{(t)}\setminus\mathcal{S}}{\arg\max}\, \boldsymbol{x}_j^{(t)\top}\boldsymbol{r}. \tag{11}$$

Then, using the updated $\mathcal{S}$, the self-expressive coefficient vector $\boldsymbol{c}_i^{*(t)}$ is found by solving the following problem:

$$\boldsymbol{c}_i^{*(t)} = \begin{cases} \arg\min_{\boldsymbol{c}_i^{(t)}} \|\boldsymbol{x}_i^{(t)} - X^{(t)}\boldsymbol{c}_i^{(t)}\|_2^2, & \text{if } i \in \mathcal{I}^{(t)}, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$$
$$\text{such that} \quad \text{supp}(\boldsymbol{c}_i^{(t)}) \subseteq \mathcal{S}, \tag{12}$$

where $\text{supp}(\boldsymbol{c}_i^{(t)})$ is the support function that returns the subgroup of the domain containing the elements not mapped to zero. $\boldsymbol{r}$ is updated using:

$$\boldsymbol{r} \leftarrow \boldsymbol{x}_i^{(t)} - X^{(t)}\boldsymbol{c}_i^{*(t)}. \tag{13}$$

This process is repeated until the number of iterations k reaches its limit s or $\boldsymbol{r}$ is smaller than the error ϵ.

To find $\boldsymbol{b}^{*(i)}$, Eq. (8) can also be solved via the OMP algorithm, as shown in Algorithm 3. The input data matrix $Y \in \mathbb{R}^{D\times T}$ is generated by Eq. (5) from $\boldsymbol{c}_i^{*(t)}$. Note that the size of Y depends on the number of subsets T and is much smaller than the number of data points. The maximum number of repetitions is T, and $\mathcal{S}$ is updated by finding the index j^* satisying

$$j^* = \underset{j\in[T]\setminus\mathcal{S}}{\arg\max}\, \boldsymbol{y}_i^{(j)\top}\boldsymbol{r}. \tag{14}$$

In addition, the weight coefficient vector $\boldsymbol{b}^{*(i)}$ and update of $\boldsymbol{r}$ are determined by solving

$$\boldsymbol{b}^{*(i)} = \underset{\boldsymbol{b}^{(i)}}{\arg\min} \|\boldsymbol{x}_i - Y\boldsymbol{b}^{(i)}\|_2^2, \quad \text{s.t. supp}(\boldsymbol{b}_i) \subseteq \mathcal{S}. \tag{15}$$

$$\boldsymbol{r} \leftarrow \boldsymbol{x}_i - Y\boldsymbol{b}_t^{*(i)}. \tag{16}$$

For clarity, we summarize the entire framework of our

Algorithm 2 OMP algorithm for finding $\boldsymbol{c}_i^{*(t)}$

Input: Data matrix $X^{(t)} \in \mathbb{R}^{D\times N}$, reference data point $\boldsymbol{x}_i^{(t)}$, maximum repetition count s, error term ϵ

Initialize $k = 0$, residual $\boldsymbol{r} = \boldsymbol{x}_i^{(t)}$, support set $\mathcal{S} = \emptyset$;
while $k < s$ and $\|\boldsymbol{r}\|_2 > \epsilon$ **do**
  Find j^* via Eq. (11);
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{j^*\}$;
  Estimate $\boldsymbol{c}_i^{*(t)}$ via Eq. (12);
  Update the residual $\boldsymbol{r}$ via Eq. (13);
  $k = k + 1$;
end while

Output: Self-expressive coefficient vector $\boldsymbol{c}_i^{*(t)}$

Algorithm 3 OMP algorithm for finding $\boldsymbol{b}^{*(i)}$

Input: Data matrix $Y \in \mathbb{R}^{D\times T}$, reference data point $\boldsymbol{x}_i$, error term ϵ

Initialize $k = 0$, residual $\boldsymbol{r} = \boldsymbol{x}_i$, support set $\mathcal{S} = \emptyset$;
while $k < T$ and $\|\boldsymbol{r}\|_2 > \epsilon$ **do**
  Find j^* via Eq. (14);
  Update $\mathcal{S} \leftarrow \mathcal{S} \cup \{j^*\}$;
  Estimate $\boldsymbol{b}^{*(i)}$ via Eq. (15);
  Update the residual $\boldsymbol{r}$ via Eq. (16);
  $k = k + 1$;
end while

Output: Weight coefficient vector $\boldsymbol{b}^{*(i)}$

Algorithm 4 Parallelizable multi-subset based sparse subspace clustering (PMSSC)

Input: Data matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$, parameters T, δ, s, ϵ

1: Compute coefficient matrix C^* via Algorithm 1;
2: Define affinity matrix $W = |C^*| + |C^{*T}|$;
3: Apply spectral clustering;

Output: Clustering results of X

proposed subspace clustering approach in Algorithm 4, calling it the *parallelizable multi-subset based sparse subspace clustering* (PMSSC) method. Given X and parameters T, δ, s, and ϵ, the optimal solution C^* can be found using Algorithm 1. We thus define the affinity matrix as $A = |C^*| + |C^{*T}|$ using the computed C^*; the final clustering results can be obtained by applying spectral clustering to A via normalized cut [19].

4 Experiments and Results

We have evaluated our approach using both synthetic data and real-world benchmark datasets.

4.1 Baselines and Evaluation Metrics.

We compare our approach to the following eight methods: SCC [25], LRR [28], thresholding-based subspace clustering (TSC) [40], low rank subspace clustering (LRSC) [41], SSSC [13], EnSC [29], SSC-OMP [10], and S^{5}C [14]. Tests for all comparative methods used provided source code, and each parameter was carefully tuned to give the best clustering accuracy. For spectral clustering, except for SCC, S^{5}C, and SSSC, we applied normalized cut [19] to the affinity matrix $A = |C| + |C^T|$. (SCC and S^{5}C have their own spectral clustering phase, while SSSC obtains clustering results from the data split into two parts). Unlike SSC-OMP, our method, which involves independent calculation for each subset, can be implemented in parallel with multi-core processing. All algorithms ran on an AMD Ryzen 7 3700x processor with 32 GB RAM. Following [10], as quantitative evaluation metrics, we evaluated each algorithm using clustering accuracy (acc: $a\%$), subspace-preserving representation error (sre: $e\%$), connectivity (conn: c), and runtime (t seconds). Clustering accuracy represents the percentage of correctly labeled data points:

$$a = \max_{\pi} \frac{100}{N} \sum_{ij} Q^{\text{est}}_{\pi(i)j} Q^{\text{true}}_{ij}, \tag{17}$$

where π is a permutation of the L cluster groups. Q^{est} and Q^{true} are the estimated labeling result and ground-truth, respectively, scoring one in the (i, j)th entry if a data point j belongs to the i-th cluster and zero otherwise. The subspace-preserving representation error indicates the average fraction of affinities formed from other subspaces in each $\boldsymbol{c}_j$,

$$e = \frac{100}{N} \sum_{j} \left(1 - \sum_{i} (\omega_{ij} |c_{ij}|) / \|\boldsymbol{c}_j\|_1 \right), \tag{18}$$

where $\omega_{ij} \in \{0, 1\}$ is the true affinity, and $\| \cdot \|_1$ returns the ℓ_1 norm. The connectivity shows the average connection of the affinity matrix with L cluster groups as follows:

$$c = \frac{1}{L} \sum_{i=1}^{L} \lambda_2^{(i)}, \tag{19}$$

where λ_2 is the second smallest eigenvalue of the normalized Laplacian for each of the L subgraph, and $\lambda_2^{(i)}$ indicates the algebraic connectivity for each cluster. If $c = 0$, then at least one subgraph is not connected [42].

4.2 Experiments on Synthetic Data

4.2.1 *Setup*

We first report experimental results on data synthesised by randomly generating five linear subspaces of $\mathbb{R}^6$ as ground-truth in an ambient space of $\mathbb{R}^9$. Each subspace contains n randomly sampled data points. To confirm the statistical results, we conducted the experiments by varying n from 100 to 4,000, so the total number N of data points varies from 500 to 20,000. We set the parameters $s = 6$, $\delta = 0.3$, and $T = 16$. The percentage of in-sample in SSSC is set to 10% of the total number of data points. All experimental results recorded on synthetic data were averaged over 50 trials.

4.2.2 *Results*

The curve for each metric is shown as a function of n in Fig. 2. We can observe from Fig. 2(a) that PMSSC outperforms SSC-OMP in terms of clustering accuracy. The difference is especially large when the density of data points on the underlying subspace is lower. This could be partly due to the fact that PMSSC succeeds in generating better connectivity than SSC-OMP (see Fig. 2(c)), and achieves lower subspace-preserving representation error (see Fig. 2(b)). On the other hand, as Fig. 2(d) shows, PMSSC is much faster with parallel implementation, which is advisable for solving problems involving large-scale data. In addition, compared to SSSC which adopts a similar sampling approach to our method, PMSSC outperforms both in clustering accuracy and runtime (using a parallel implementation).

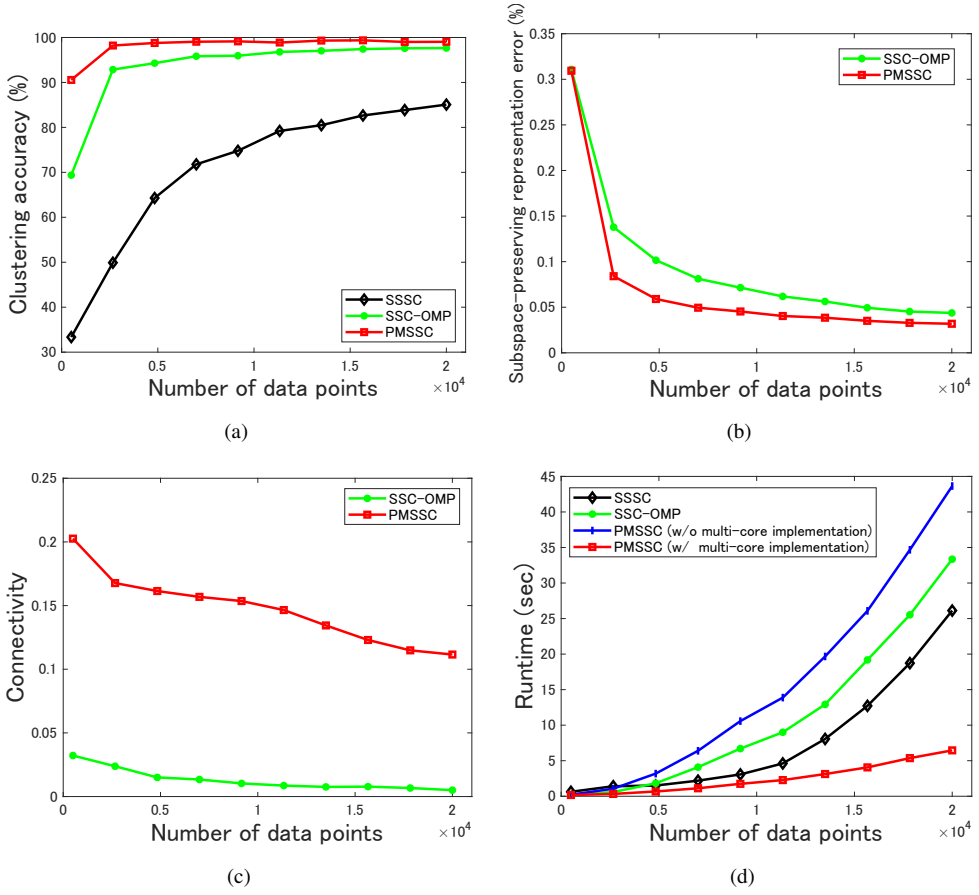


Fig. 2 Comparison of PMSSC, SSC-OMP, and SSSC on synthetic data in terms of (a) clustering accuracy, (b) subspace-preserving representation error, (c) connectivity, and (d) runtime. For SSSC, only clustering accuracy and runtime are shown as SSSC does not generate the self-expressive coefficient matrix and affinity matrix.

Table 1 Parameters (s, δ, and T) used in PMSSC for benchmark datasets.

Datasets	Ex. Yale B	ORL	GTSRB	BBCSport	MNIST4000	MNIST10000	MNIST	EMNIST	CIFAR-10
s	5	5	3	3	10	10	10	10	3
δ	0.6	0.6	0.2	0.4	0.3	0.2	0.1	0.2	0.2
T	6	11	8	15	7	10	19	12	18

4.3 Experiments on Benchmark Datasets for Real-world Applications

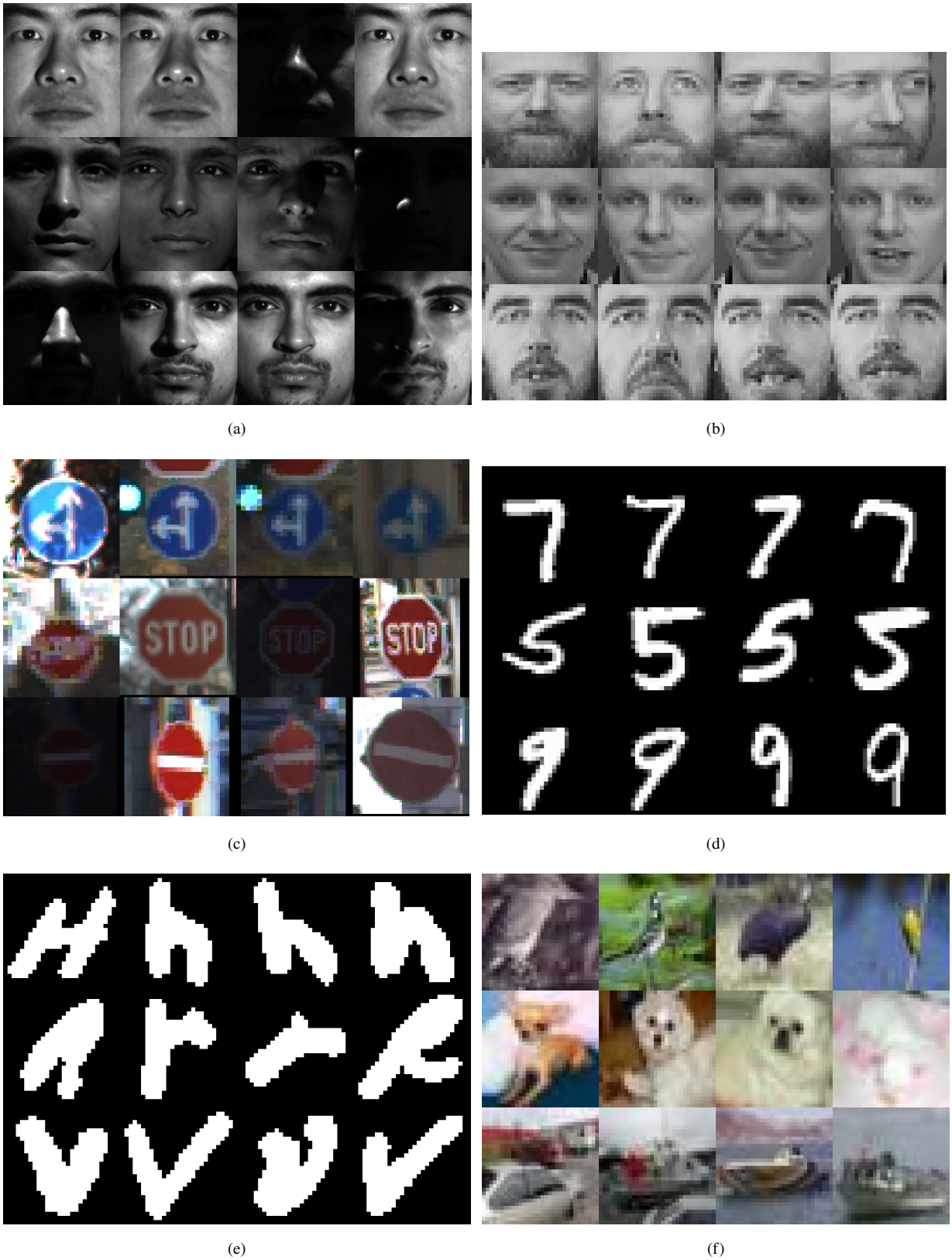
4.3.1 Setup

We conducted experiments on five benchmark datasets: Extended Yale B [43] and ORL [44] for face clustering, BBCSport [45] for text document clustering, German Traffic Sign Recognition Benchmark (GTSRB) [46] for street sign clustering, Modified National Institute of Standards and Technology database (MNIST) [47] and Extended MNIST (EMNIST) [48] for handwritten character clustering, and Canadian Institute For Advanced Research (CIFAR-10) [49] for object clustering. Parameter settings used for our method in these experiments are shown in Table 1. Since the sparsity s in PMSSC and SSC-OMP is related to the intrinsic dimensionality of the subspace, it is manually determined to be close to the dimensionality of the subspaces. For sampling rate δ, we picked a smaller δ for a larger dataset. For the number of subsets T, we adopted a value that takes into account the trade-off between runtime and clustering accuracy. All experimental results recorded on all benchmark datasets are averaged over 10 trials. Details of each benchmark dataset and the corresponding clustering results are now given.

4.3.2 Extended Yale B

Extended Yale B contains 2,432 facial images in 38 classes; see Fig. 3(a). In this experiment, following [9], we concatenated the pixels of each image resized to 48×42, and used the 2016-dimensional vector as input data.

The results on Extended Yale B are shown in Table 2. In each



(a) (b)

(c) (d)

(e) (f)

Fig. 3 Visual examples for datasets: (a) Extended Yale B, (b) ORL, (c) GTSRB, (d) MNIST, (e) EMNIST-Letters, and (f) CIFAR-10.

Table 2 Comparative results on Extended Yale B; '-' indicates the metric cannot be computed.

Method	Extended Yale B			
	acc ($a\%$)	sre ($e\%$)	conn (c)	t (sec.)
SCC	9.54	-	-	293.01
LRR	37.58	97.44	**0.8175**	219.91
TSC	52.40	-	0.0014	10.46
LRSC	56.71	91.64	0.4360	4.37
SSSC	49.77	-	-	18.22
EnSC	55.76	**18.90**	0.0395	4.57
SSC-OMP	73.82	20.07	0.0364	**1.27**
S^5C	62.99	58.74	0.2238	952.26
PMSSC	**80.24**	22.35	0.0858	2.66

Table 3 Comparative results on ORL.

Method	ORL			
	acc ($a\%$)	sre ($e\%$)	conn (c)	t (sec.)
SCC	16.40	-	-	89.45
LRR	32.98	97.70	**0.8394**	3.57
TSC	68.03	-	0.0992	0.72
LRSC	43.12	93.72	0.5248	0.24
SSSC	65.12	-	-	1.78
EnSC	70.03	**32.46**	0.1825	0.65
SSC-OMP	60.12	34.14	0.0770	**0.12**
S^5C	69.48	63.26	0.3868	54.28
PMSSC	**74.45**	40.97	0.1708	0.51

column, the best result is shown in bold, and the second-best result is underlined. They confirm that PMSSC yields the best clustering accuracy, and improves the clustering accuracy over SSC-OMP by 6.42%. Although the subspace-preserving error and runtime are slightly lower than SSC-OMP, the connectivity is greatly improved compared to SSC-OMP, leading to a better clustering accuracy. LRR, LRSC, and S^5C have good connectivity, but poor subspace-preserving errors result in low clustering accuracy.

4.3.3 *ORL*

ORL contains 400 facial images in 40 classes, as shown in Fig. 3(b). In this experiment, following [50], we concatenate the pixels of each image resized to 32×32, and use a 1024-dimensional vector as input data. Compared to Extended Yale B, ORL is a more difficult problem setting for subspace clustering because the density of data lying near the same subspace (10 vs. 64) is lower due to the small number of images of each subject, and the subspaces have more non-linearity due to changes in facial expressions and details.

The results for ORL are listed in Table 3. We can again observe that PMSSC achieves the best clustering accuracy, and improves the connectivity compared to SSC-OMP. However, since PMSSC does not incorporate nonlinear constraints, the subspace-preserving error does not improve along with the improvement of the connectivity.

Table 4 Comparative results on GTSRB.

Method	GTSRB			
	acc ($a\%$)	sre ($e\%$)	conn (c)	t (sec.)
SCC	59.68	-	-	84.26
LRR	27.87	86.64	0.4255	725.18
TSC	56.36	-	0.0016	242.62
LRSC	83.97	80.14	**0.6056**	12.89
SSSC	88.03	-	-	16.86
EnSC	85.92	**0.59**	0.0065	10.77
SSC-OMP	81.28	5.38	0.0211	3.72
S^5C	61.60	80.99	0.5941	422.35
PMSSC	**91.57**	7.69	0.0434	**3.40**

4.3.4 *GTSRB*

GTSRB contains over 50,000 street sign images in 43 classes; see Fig. 3(c). Following [33], we preprocess the dataset represented by a 1568-dimensional HOG feature to get an imbalanced dataset of the 500-dimensional vectors with 12,390 samples in 14 classes.

The results on GTSRB are reported in Table 4. Again PMSSC yields the best clustering accuracy and runtime, and improves the clustering accuracy roughly by 10% compared to SSC-OMP. In particular, PMSSC has both good subspace-preserving error and connectivity. While EnSC and SSSC also achieve competitive clustering accuracy, their computational costs are much higher.

4.3.5 *BBCSport*

BBCSport contains 737 documents in five classes. The data provided by the database has been preprocessed by stemming, stop-word removal, and low term frequency filtering. In this experiment, we reduced the dimensionality of feature vectors to 500 by PCA.

The results on BBCSport are summarized in Table 5. We can observe that PMSSC yields the second best clustering accuracy and subspace-preserving error. LRSC yields the best clustering accuracy due to good connectivity. For small-scale datasets such as BBCSport and ORL, the speed of PMSSC is slightly lower than for SSC-OMP because the advantage of reducing data size by sampling multiple subsets is diminished.

4.3.6 *MNIST and EMNIST-Letters*

MNIST contains 70,000 images of handwritten digits (0–9), while EMNIST-Letters contains 145,600 images of handwritten characters in 26 classes, as shown in Fig. 3(d,e). In our experiments, following [34], we generate MNIST4000 and MNIST10000, which are produced by randomly sampling 400 and 1,000 images per class of digit, respectively. Each image is represented as a 3472-dimensional feature vector by using the scattering convolution network [51], and its dimensionality reduced to 500 by PCA.

Table 5 Comparative results on BBCSport.

Method	BBCSport			
	acc (a%)	sre (e%)	conn (c)	t (sec.)
SCC	23.12	-	-	3.60
LRR	71.37	76.26	**0.7744**	7.59
TSC	73.95	-	0.0053	0.36
LRSC	**89.53**	66.38	0.5997	0.18
SSSC	50.24	-	-	0.26
EnSC	59.48	**11.43**	0.0243	0.61
SSC-OMP	69.85	15.96	0.0393	**0.10**
S^{5}C	55.90	65.78	0.5434	17.99
PMSSC	81.71	14.36	0.0509	0.47

The results on MNIST and EMNIST-Letters are summarized in Tables 6–8. We can observe that PMSSC yields the best clustering accuracy on MNIST4000, MNIST10000, and EMNIST-Letters. In particular, PMSSC is remarkably faster than the comparative methods on MNIST70000 and EMNIST-Letters. In the case of MNIST70000, EnSC yields the best clustering accuracy and subspace-preserving error but its computational cost is high. Similarly, S^{5}C can achieve good connectivity, but is very slow.

4.3.7 CIFAR-10

CIFAR-10 includes 60,000 general objects in 10 classes, as illustrated in Fig. 3(f). Following [52], we employ the feature representations extracted by MCR^2 [53], which learns a union of low-dimensional subspaces representation via self-supervised learning. Each feature is represented by a 128-dimensional feature vector, further normalized to have unit ℓ_2 norm.

The comparative results on CIFAR-10 are summarized in Table 9. It can be observed that our method outperforms others in terms of the runtime, while the clustering accuracy is competitive. However, as with SSC-OMP, we see that the connectivity is lower than for S^{5}C, which uses ℓ_1 norm regularization.

4.3.8 Summary

Overall, our proposed method becomes significantly faster as the amount of input data increases. In addition, it achieves good clustering accuracy and connectivity, and provides subspace-preserving errors comparable to those of the comparative algorithms.

4.4 Analysis

4.4.1 Multi-subset Based Self-Expressive Model

Since our approach aims to minimize the self-expressive residual by the weight coefficient vector $\boldsymbol{b}^*$ solved in Algorithm 3, we show the mean self-expressive residual of data points represented by the coefficient vectors in Fig. 4.

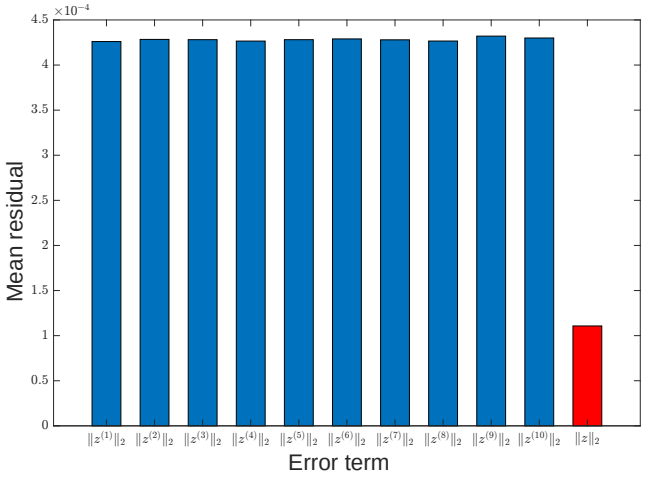


Fig. 4 Comparative results in terms of the mean residuals over data points represented by the self-expressive models with different subsets. Blue bars represent each single t-th subset, while the red bar is computed using multiple subsets.

This experiment was performed on synthetic data, and we fixed $T = 10$ and $\delta = 0.3$. Each blue bar indicates the mean self-expressive residual of the data points represented by Eq. (5), computed as

$$\|\boldsymbol{z}^{(t)}\|_2 = \frac{1}{\lceil \delta N \rceil} \sum_{i=1}^{N} \|\boldsymbol{x}_i^{(t)} - X^{(t)} \boldsymbol{c}_i^{*(t)}\|_2. \tag{20}$$

The red bar indicates the mean self-expressive residual of the data points represented by Eq. (9), computed as:

$$\|\boldsymbol{z}\|_2 = \frac{1}{N} \sum_{i=1}^{N} \|(\boldsymbol{x}_i - X \boldsymbol{c}_i^*)\|_2. \tag{21}$$

We can clearly observe that the mean self-expressive residual of PMS has lower error than every single subset. To highlight the benefit of $\boldsymbol{b}^*$, we made a comparison to a variant of our approach, named PMSSC(avg), which replaced $\boldsymbol{b}^*$ by a simple average operation: in PMSSC(avg), Eq. (10) is replaced by

$$\boldsymbol{c}_i^* = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{c}_i^{*(t)}. \tag{22}$$

We performed experiments on synthetic data using the same setup as for Fig. 2 and present the results in Fig. 5. As can be seen, incorporating $\boldsymbol{b}^*$ improves clustering performance; in particular, the subspace-preserving representation error is significantly reduced. These experiments indicate that the weight coefficient vector $\boldsymbol{b}^*$ contributes to improving self-expressiveness.

4.4.2 Selection of Parameters

We performed multiple experiments on the GTSRB dataset with various choices of hyperparameters (T, δ) to evaluate the sensitivity of our approach to parameter choice. Changes in clustering accuracy, subspace-preserving representation

Table 6 Comparative results on MNIST4000 and MNIST10000.

Method	MNIST4000				MNIST10000			
	acc (a%)	sre (e%)	conn (c)	t (sec.)	acc (a%)	sre (e%)	conn (c)	t (sec.)
SCC	67.45	-	-	5.93	70.43	-	-	11.67
LRR	78.49	90.21	**0.8979**	43.03	77.53	90.60	**0.8818**	396.45
TSC	79.57	-	0.0009	11.76	80.62	-	0.0005	132.08
LRSC	81.23	75.67	0.5984	1.61	80.86	77.30	0.5983	7.58
SSSC	70.73	-	-	3.11	84.32	-	-	13.20
EnSC	89.08	**21.14**	0.1174	12.71	88.24	**17.34**	0.0975	35.63
SSC-OMP	91.49	34.69	0.1329	1.61	91.40	32.23	0.1169	6.44
S^{5}C	81.52	66.28	0.4476	277.93	79.30	66.23	0.4466	683.65
PMSSC	**92.85**	38.27	0.1944	**1.42**	**93.57**	36.43	0.1817	**4.55**

Table 7 Comparative results on MNIST; 'M' indicates that 32 GB memory was exhausted.

Method	MNIST70000			
	acc (a%)	sre (e%)	conn (c)	t (sec.)
SCC	69.08	-	-	388.00
LRR	M	-	-	-
TSC	M	-	-	-
LRSC	M	-	-	-
SSSC	81.57	-	-	303.28
EnSC	**93.79**	**11.26**	0.0596	408.62
SSC-OMP	82.83	28.57	0.0830	248.50
S^{5}C	72.99	66.87	**0.4437**	4953.28
PMSSC	84.45	32.63	0.1148	**65.08**

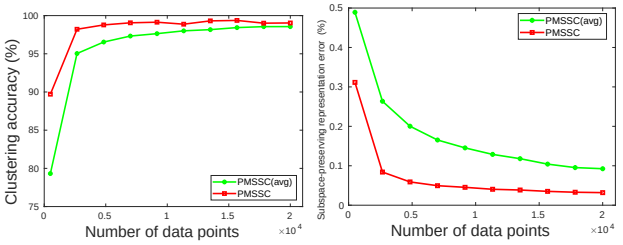
Table 8 Comparative results on EMNIST-Letters; 'M' indicates that 32 GB memory was exhausted.

Method	EMNIST-Letters			
	acc (a%)	sre (e%)	conn (c)	t (sec.)
SCC	M	-	-	-
LRR	M	-	-	-
TSC	M	-	-	-
LRSC	M	-	-	-
SSSC	60.62	-	-	1538.46
EnSC	64.15	**26.20**	0.0086	1575.46
SSC-OMP	58.71	43.93	0.0000	1214.31
S^{5}C	60.01	83.37	**0.3517**	15698.90
PMSSC	**66.52**	46.76	0.0019	**638.03**

error, connectivity, and runtime when varying each parameter are illustrated in Fig. 6. We can confirm that high clustering accuracy and low subspace-preserving representation error are maintained in most cases, except when both T and δ are extremely small. This implies that the affinity matrix constructed by PMSSC provides subspace-preserving representations at most data points. We can also see that the connectivity improves as the number of subsets T increases, because the affinity matrix contains at most sTN nonzero entries in OMP optimization. Considering runtime, a practical choice of parameters is to increase T for small values of δ, and decrease T for large values of δ. In addition, time taken can be kept low by picking a small value of δ for large-scale datasets.

Table 9 Comparative results on CIFAR-10 where 'M' means that the memory limitation of 32G is exceeded.

Method	CIFAR-10			
	acc (a%)	sre (e%)	conn (c)	t (sec.)
SCC	37.10	-	-	196.40
LRR	M	-	-	-
TSC	M	-	-	-
LRSC	M	-	-	-
SSSC	63.80	-	-	74.36
EnSC	61.79	**22.60**	0.0000	178.22
SSC-OMP	40.86	24.92	0.0000	63.58
S^{5}C	**64.52**	46.35	**0.2314**	2338.55
PMSSC	63.52	26.41	0.0000	**29.60**



(a) (b)

Fig. 5 Benefit of using $\boldsymbol{b}^*$ in PMSSC in terms of (a) clustering accuracy and (b) subspace-preserving representation error, for synthetic data. Red: using $\boldsymbol{b}^*$. Green: using simple averaging.

4.4.3 Sampling Technique

Our approach adopts weighted random sampling to generate the subset data matrix $X^{(t)}$. To analyze the effect of sampling methods on our approach, we compared weighted random sampling to random sampling with uniform weights. The experimental settings used for synthetic data follow those in Fig. 2. Fig. 7 shows the clustering accuracy and connectivity as functions of n. Obviously, weighted random sampling outperforms random sampling in terms of both clustering accuracy and connectivity. In particular, as the density of data points increases, the connectivity of the method with random sampling becomes zero, because imbalanced sampling leads to a disconnected subgraph in an affinity graph.

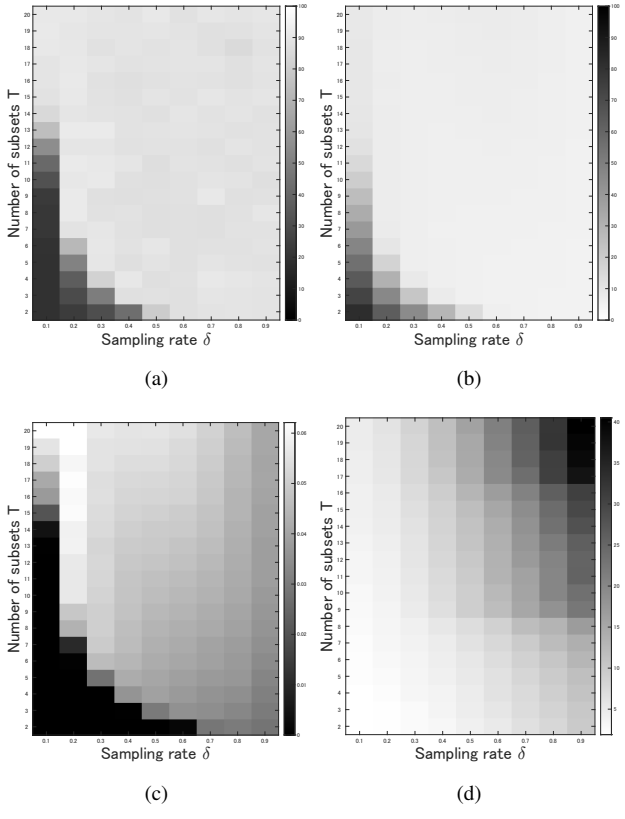


Fig. 6 Effects of varying parameters δ and T (GTSRB dataset): (a) clustering accuracy, (b) subspace-preserving representation error, (c) connectivity, and (d) runtime.

4.4.4 *Computational Complexity*

Algorithms 2 and 3 for affinity matrix construction consume most of the processing time. In Algorithm 2, the computational complexity for finding the self-expressive coefficient vector $\boldsymbol{c}_i^{*(t)}$ requires time $\mathcal{O}(Ds\lceil\delta N\rceil)$. In Algorithm 3, the computational complexity for finding the weight coefficient vector $\boldsymbol{b}^{*(i)}$ requires $\mathcal{O}(DT^2)$. Because these two algorithms are performed on N data points, the computational complexity of PMS requires at least time $\mathcal{O}(N(TDs\lceil\delta N\rceil + DT^2))$. However, processing T subsets (the part taking $\mathcal{O}(TDs\lceil\delta N\rceil)$) can be performed in parallel, which reduces the computation time compared to methods that directly deal with the whole dataset. Fig. 2(d) supports this analysis.

5 Conclusions

We have proposed a parallelizable multi-subset based self-expressive model for subspace clustering. A representation of the input data is formulated by combining the solutions of small optimization problems with respect to multiple subsets generated by data sampling. We have shown that this strategy can significantly improve speed with a multi-core approach that can be easily implemented, especially for large-scale data. Moreover, it has been verified that combining multiple subsets can reduce the self-expressive residuals of data compared to a single subset. Extensive experiments on synthetic data and real-world datasets have demonstrated the efficiency and effectiveness of our approach. As a limitation, our method is still unable to handle nonlinear subspaces due to the problem setting. In future, we would like to design a self-expressive model that can handle nonlinear subspaces, with the help of modeling capabilities from neural network architectures.

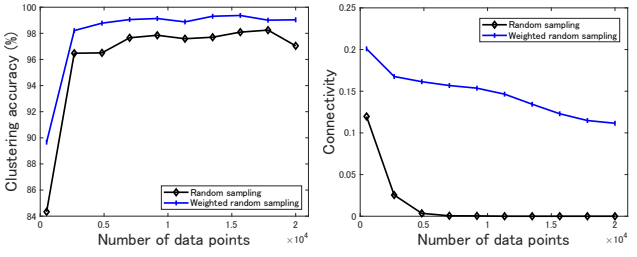


Fig. 7 Effect of sampling method in our approach, for synthetic data: (a) clustering accuracy and (b) connectivity. Blue: weighted random sampling. Black: uniform random smapling.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP20K19568.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Vidal R. Subspace clustering. *IEEE Signal Processing Magazine*, 2011, 28(2): 52–68.

[2] Hotta K, Xie H, Zhang C. Affine subspace clustering with nearest subspace neighbor. In *International Workshop on Advanced Imaging Technology (IWAIT) 2021*, volume 11766, 2021, 117661C.

[3] Zhang C. Energy Minimization over m-Branched Enumeration for Generalized Linear Subspace Clustering. *IEICE Transactions on Information and Systems*, 2019, 102(12): 2485–2492.

[4] Yang AY, Wright J, Ma Y, Sastry SS. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 2008, 110(2): 212–225.

[5] Vidal R, Tron R, Hartley R. Multiframe motion segmentation with missing data using PowerFactorization and GPCA. *International Journal of Computer Vision*, 2008, 79(1): 85–105.

[6] Tierney S, Gao J, Guo Y. Subspace clustering for sequential data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, 1019–1026.

[7] Zhang C, Lu X, Hotta K, Yang X. G2MF-WA: Geometric multi-model fitting with weakly annotated data. *Computational Visual Media*, 2020, 6: 135–145.

[8] Elhamifar E, Vidal R. Sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 6, 2009, 2790–2797.

[9] Elhamifar E, Vidal R. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013, 35(11): 2765–2781.

[10] You C, Robinson D, Vidal R. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 3918–3927.

[11] Guo Y, Tierney S, Gao J. Efficient sparse subspace clustering by nearest neighbour filtering. *Signal Processing*, 2021, 185: 108082.

[12] You C, Li C, Robinson D, Vidal R. Self-Representation Based Unsupervised Exemplar Selection in a Union of Subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[13] Peng X, Zhang L, Yi Z. Scalable sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, 430–437.

[14] Matsushima S, Brbic M. Selective sampling-based scalable sparse subspace clustering. *Advances in Neural Information Processing Systems*, 2019, 32: 12416–12425.

[15] Tseng P. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 2000, 105(1): 249–252.

[16] Zhang T, Szlam A, Lerman G. Median k-flats for hybrid linear modeling with many outliers. In *Conference on Computer Vision Workshops*, 2009, 234–241.

[17] Lipor J, Hong D, Tan YS, Balzano L. Subspace clustering using ensembles of k-subspaces. *Information and Inference: A Journal of the IMA*, 2021, 10(1): 73–107.

[18] Lane C, Haeffele B, Vidal R. Adaptive online k-subspaces with cooperative re-initialization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, 678–688.

[19] Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000, 22(8): 888–905.

[20] Von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing*, 2007, 17(4): 395–416.

[21] Lu C, Feng J, Lin Z, Mei T, Yan S. Subspace Clustering by Block Diagonal Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018, 41(2): 487–501.

[22] Dong W, Wu XJ, Kittler J, Yin HF. Sparse subspace clustering via nonconvex approximation. *Pattern Analysis and Applications*, 2019, 22(1): 165–176.

[23] Hotta K, Xie H, Zhang C. Candidate Subspace Screening for Linear Subspace Clustering with Energy Minimization. In *Irish Machine Vision and Image Processing Conference*, 2020, 125–128.

[24] Yan J, Pollefeys M. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, 2006, 94–106.

[25] Chen G, Lerman G. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 2009, 81(3): 317–330.

[26] Donoho DL. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 2006, 59(6): 797–829.

[27] Lu CY, Min H, Zhao ZQ, Zhu L, Huang DS, Yan S. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, 2012, 347–360.

[28] Liu G, Lin Z, Yu Y, et al.. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, 663–670.

[29] You C, Li CG, Robinson DP, Vidal R. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 3928–3937.

[30] Ji P, Salzmann M, Li H. Efficient dense subspace clustering. In *IEEE Winter Conference on Applications of Computer Vision*, 2014, 461–468.

[31] Dyer EL, Sankaranarayanan AC, Baraniuk RG. Greedy feature selection for subspace clustering. *The Journal of Machine Learning Research*, 2013, 14(1): 2487–2517.

[32] Nasihatkon B, Hartley R. Graph connectivity in sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011, 2137–2144.

[33] You C, Li C, Robinson DP, Vidal R. Scalable exemplar-based subspace clustering on class-imbalanced data. In *Proceedings of the European Conference on Computer Vision*, 2018, 67–83.

[34] Chen Y, Li CG, You C. Stochastic sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, 4155–4164.

[35] You C, Donnat C, Robinson DP, Vidal R. A divide-and-conquer framework for large-scale subspace clustering. In *Proceedings of 50th Asilomar Conference on Signals, Systems and Computers*, 2016, 1014–1018.

[36] Davenport MA, Wakin MB. Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory*, 2010, 56(9): 4395–4401.

[37] Tropp JA. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 2004, 50(10): 2231–2242.

[38] Pati YC, Rezaiifar R, Krishnaprasad PS. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of Asilomar Conference on Signals, Systems and Computers*, 1993, 40–44.

[39] Wong CK, Easton MC. An efficient method for weighted

sampling without replacement. *SIAM Journal on Computing*, 1980, 9(1): 111–113.

[40] Heckel R, Bölcskei H. Subspace clustering via thresholding and spectral clustering. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, 3263–3267.

[41] Vidal R, Favaro P. Low rank subspace clustering (LRSC). *Pattern Recognition Letters*, 2014, 43: 47–61.

[42] Chung FR, Graham FC. Spectral graph theory, 1997.

[43] Lee KC, Ho J, Kriegman DJ. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, 27(5): 684–698.

[44] Samaria FS, Harter AC. Parameterisation of a stochastic model for human face identification. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 1994, 138–142.

[45] Greene D, Cunningham P. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *Proc. 23rd International Conference on Machine learning*, 2006, 377–384.

[46] Stallkamp J, Schlipsing M, Salmen J, Igel C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012, 32: 323–332.

[47] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324.

[48] Cohen G, Afshar S, Tapson J, Van Schaik A. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks*, 2017, 2921–2926.

[49] Krizhevsky A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

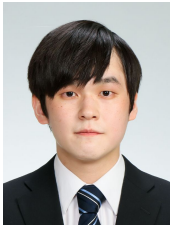
[50] Cai D, He X, Hu Y, Han J, Huang T. Learning a spatially smooth subspace for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, 1–7.

[51] Bruna J, Mallat S. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013, 35(8): 1872–1886.

[52] Zhang S, You C, Vidal R, Li CG. Learning a self-expressive network for subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021, 12393–12403.

[53] Yu Y, Chan KHR, You C, Song C, Ma Y. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 2020, 33: 9422–9434.

Author biographies



Katsuya Hotta received a B.E. degree in 2017 and is now pursuing a Ph.D. degree at the University of Fukui, Japan. His research interests lie in unsupervised learning (e.g. subspace clustering), computer vision (e.g. anomaly detection), and other aspects of machine learning.

Takuya Akashi received a Ph.D. in System Design Engineering from the University of Tokushima in 2006. Since April 2009, he has been at the Department of Electrical Engineering, Electronics and Computer Science, Iwate University. In 2015, he was a visiting associate at California Institute of Technology. He is currently an associate professor at Iwate University. His research interests include evolutionary algorithms, image processing and human sensing. He is a member of the IEEE, RISP, IEICE, and IEEJ.

Shogo Tokai received B.S., M.S., and Ph.D. degrees from Nagoya University, in 1991, 1993, and 1996 respectively. He is currently a professor in the Department of Engineering, University of Fukui. His current research includes computer graphics, computer vision, and multiple-view video processing. He is a member of ITE, IPSJ, and IEICE.

Chao Zhang received his Ph.D. degree from Iwate University in 2017. He is now a senior lecturer in the Department of Engineering, University of Fukui. His research interests include computer vision, computer graphics, and evolutionary computing, mainly focused on applying optimization methods to solve visual computing problems.