Pattern Recognition 172 (2026) 112618

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Pseudo labels approach to interpretable self-guided subspace clustering

Ivica Kopriva

Division of Computing and Data Science, Ruđer Bošković Institute, Zagreb, Croatia

A R T I C L E I N F O

Dataset link: https://github.com/ikopriva/LFSGSC

Keywords:
Subspace clustering
Hyperparameter optimization
Pseudo labels
Interpretability

A B S T R A C T

Majority subspace clustering (SC) algorithms depend on one or more hyperparameters that need to be tuned for the SC algorithms to achieve high clustering performance. This is often performed using grid-search, assuming that held out set is available. In some domains, such as medicine, this assumption does not hold true in many cases. To address this problem, we propose an approach to label-independent hyperparameter optimization by applying the SC algorithm to the data and use the resulting cluster assignments as pseudo-labels to compute clustering quality metrics (e.g., accuracy (ACC) or normalized mutual information (NMI)) across a predefined hyperparameter grid. Assuming that ACC (or NMI) is a smooth function of hyperparameter values, it is possible to select subintervals of hyperparameters, which are then iteratively further split into halves or thirds until a relative error criterion is satisfied. In principle, the hyperparameters of any SC algorithm can be tuned using the proposed method. We demonstrate this approach on five single-view SC algorithms and two multi-view SC algorithms, comparing the achieved performance with their oracle versions across six datasets for single-view algorithms and three datasets for multi-view algorithms. The proposed method typically achieves clustering performance that is up to 7 % lower than that of the oracle versions. We also enhance the interpretability of the proposed method by visualizing subspace bases, estimated from the computed clustering partitions. This aids in the initial selection of the hyperparameter search space.

1. Introduction

Clustering is a fundamental problem in unsupervised learning. It has numerous applications, including medical image analysis [1], single-cell transcriptomics [2], pattern recognition and speaker identification [3], among others. Due to the complex shapes of samples spaces, distance-based clustering algorithms often struggle to cluster data accurately in the original ambient domain. However, if a high-quality data-affinity matrix can be estimated, spectral methods can achieve high clustering performance [4]. Subspace clustering (SC) algorithms focus on learning a data affinity matrix under the assumption that the data are generated by a union-of-linear subspaces [5]. Representative SC algorithms are based on self-expressive model, where sparsity constraint is imposed on representation matrix in [5], and combination of low-rank and sparsity constraint in [6]. The least squares regression (LSR) SC algorithm in [7] yields analytical solution for the representation matrix. Because contemporary data are often recorded across multiple modalities or represented by various multiple features, multi-view extensions of SC algorithms have also been proposed in [8]. In particular, to avoid problems associated with hyperparameter tuning, researchers in [9] proposed a parameter-free multi-view SC algorithm. To address the large-scale SC problem, researchers in [10] developed multi-view algorithm with linear complexity. However, real world data do not always originate from linear subspaces. To address this issue, SC algorithms can be formulated in a Reproducing Kernel Hilbert Space (RKHS), also known as the feature space, [11]. An alternative to kernel-based SC is the graph filtering approach [12]. As discussed in [12], graph filtering smooths the graph, removes noise, and iteratively incorporates graph similarity into features. This process can make data separable in the graph-filtering domain, even if they are not separable in the original space.

Although the SC algorithms cited above, as well as many related ones, exhibit excellent clustering performance on benchmark datasets, they involve one or more hyperparameters. Hyperparameter optimization (HPO) in these algorithms is primarily based on external cluster quality metrics, such as clustering error, which require a certain amount of labeled data. Although SC algorithms are designed to operate in a purely unsupervised manner, it is often assumed in practice that a held-out set with labeled samples is available [13]. However, real-world clustering tasks frequently lack label information to aid in hyperparameter selection [14]. For instance, in the medical field, the number of labeled data is limited, and human annotation is both time consuming

E-mail address: ikopriva@irb.hr.

https://doi.org/10.1016/j.patcog.2025.112618
Received 28 January 2025; Received in revised form 19 June 2025; Accepted 14 October 2025
Available online 15 October 2025

and expensive, even though a large number of unlabeled data is available [15]. Consequently, there is a growing interest in self-supervised learning algorithms [15] and self-taught algorithms [16]. They aim to learn useful features from a large number of unlabeled instances [15] or to guide themselves through learning [16]. One approach to label independent learning for SC algorithms involves using internal clustering quality metrics [17]. However, it has been noted that existing metrics for clustering quality are not suitable for evaluating the internal clustering quality of union-of-linear-subspaces models [18]. Extensive validation in [17] has demonstrated that the *K*-subspaces (KSS) cost used in the KSS algorithm [19] and the Calinski-Harabasz (CS) index are effective for hyperparameters selection when the number of clusters is known in advance. These metrics require three inputs: the data, the estimated clusters and set of subspace dimensions. The last input might be challenging to provide when working with datasets in new application domains.

It has been noted in [20] that the performance of machine learning (ML) algorithms is largely determined by the hyperparameter settings used. The main challenge is that hyperparameters must be tuned for each specific ML problem to achieve optimal performance. Consequently, HPO, which aims to find the best configuration for ML tasks, is a significant area of research topics in the ML community. The most commonly used strategy is search-based, where a predefined search space is used to find the optimal hyperparameter values for a given ML algorithm [21]. It is also used in SC algorithms such as [5] and [6]. This approach is computationally intensive and requires held-out set to evaluate performance for different hyperparameter values. An efficient alternative is meta-learning [22,23], which uses previous evaluations from historical datasets to predict desirable hyperparameters for new task. Traditionally, meta-learning has worked well for hyperparameters organized as vectors. However, researchers in [20] proposed to organize multiple hyperparameters as tensors and to formulate the interpolation of optimal hyperparameter values as a low-rank tensor completion (LRTC) problem [23]. This approach assumes that the selected performance metric is a smooth function of the hyperparameters, allowing for interpolation of optimal values from historic evaluations through solving the LRTC problem. A critical assumption of this approach is the availability of previous historic evaluations. In rapidly evolving domains, such as various medical imaging modalities, where new data are generated frequently, it is often unrealistic to rely on such historical evaluations assumption.

To address the hyperparameter tuning challenges outlined above, we propose a new pseudo labels based HPO strategy for derived SC algorithms. This method is based on clustering quality metrics such as accuracy (ACC) or normalized mutual information (NMI). However, instead of using external (hard) labels, ACC and NMI in our approach are calculated from pseudo-labels generated by the SC algorithm itself. Similar to the grid-search approach for HPO, we define a search space for selected SC algorithm where the optimal hyperparameters are expected to reside. However, this space can be less dense compared to an exhaustive greed search. In our approach, as in [20], we also assume the performance metric is a smooth function of the hyperparameters. Accordingly, we compute ACC or NMI between pseudo-labels generated by neighboring hyperparameter values. Based on the smoothness assumption, we subdivide hyperparameter intervals into smaller sections, which are further split into halves or thirds, and SC algorithm generates pseudo-labels for these interpolated values. This process is repeated iteratively until a relative error criterion is met. Thus, our HPO approach allows SC algorithms to be tuned in a label-independent manner, enabling their application in new domains where labeled data for HPO is unavailable. Furthermore, our approach complements an existing avenue of research related to development of SC algorithms that are free of hyperparameters such as [9]. In other words, our approach is proposed for label-free self-guided (LFSG) hyperparameter tuning of existing SC algorithms that, in the spirit of self-supervised learning [15] and self-taught learning [16], is self-guided. Fig. 1 illustrates our approach to HPO using as an example the least squares regression (LSR) SC algorithm [7]. MATLAB code of the proposed approach to Label-Free Self-Guided Subspace Clustering (LFSGSC) is available at https://github.com/ikopriva/LFSGSC.

In fields like medicine, achieving high diagnostic performance often requires the use of highly complex models whose decision-making processes are challenging to interpret and explain [24]. When decisions involve high stakes it becomes crucial to provide explanations for an algorithm's predictions. To the best of our knowledge, SC algorithms also face this issue of interpretability. In response, we propose a method to interpret clustering results from LFSGSC algorithms by visualizing subspace bases estimated from the obtained clustering partitions. If experts judge the visualization quality to be inadequate, the initial hyperparameter search space can be refined, and the algorithm restarted.

Our contributions in this paper are summarized as follows:

- We propose a pseudo-labels approach to HPO for existing SC algorithms. This approach uses clustering quality metrics such as ACC or NMI. However, rather than relying on actual labels, ACC or NMI are computed from pseudo-labels generated by the selected SC algorithm within a predefined search space, where the optimal hyperparameter values are expected to reside. Our approach assumes that ACC and NMI are smooth functions of hyperparameters, allowing for the

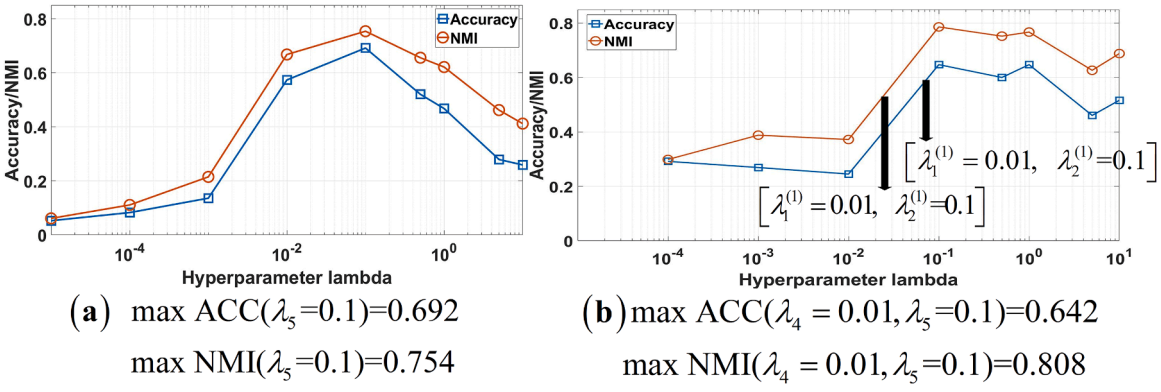


Fig. 1. Illustration of proposed approach to label-free HPO for LSR SC algorithm [9], that depends on one hyperparameter λ, see eq. (6), on Extended YaleB dataset. (**a**) Accuracy and NMI at $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 5, 10\}$ calculated with external (hard) labels. Both maximal accuracy and maximal NMI occur at $\lambda_5 = 0.1$. (**b**) Accuracy and NMI computed between pseudo-labels generated by the same grid values of λ as in (**a**). Maximal accuracy occurs between $\lambda_4 = 0.01$ and $\lambda_5 = 1$. Consequently, new iteration "1" begins on the interval with borders $\lambda_1^{(1)} = \lambda_4 = 0.01$ and $\lambda_2^{(1)} = \lambda_5 = 0.1$. The interval is further subdivided into thirds giving $\{0.01, 0.04, 0.07, 0.1\}$ and the process is repeated until convergence is reached. Regarding NMI, the process is analogous to the accuracy metric.

selection of the subintervals where ACC and NMI values are maximized. Our approach allows for the re-use of existing SC algorithms in domains where held-out set with labeled data is unavailable.

- We propose a method to enhance interpretability of LFSGSC algorithms by visualizing subspace bases, estimated from the computed clustering partitions. If the visualization quality is deemed insufficient, the initial hyperparameter search space can be adjusted, and the algorithm restarted.
- We evaluate the proposed pseudo-labels approach to HPO on several carefully selected SC algorithms and compare the resulting performance with the oracle versions of the same algorithms. Specifically, we demonstrate the effectiveness of our approach on linear single-view and multi-view SC algorithms, as well as on some nonlinear, kernel-based and graph-filtering-based versions. As shown, our approach achieves performance close to that of the oracle versions of the same SC algorithms.
- As side contributions, we formulate graph-filtering version of the LSR SC algorithm in Algorithm 1, and out-of-sample extension of kernel-based version of the LSR SC algorithm in section 3.4.

2. Background and related work

2.1. Subspace clustering

We assume that the columns of the matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$ represent data points drawn from a union of linear subspaces $\cup_{i=1}^{C} S_i$ of unknown dimensions $\{d_i = \dim(S_i)\}_{i=1}^{C}$ in $\mathbb{R}^D$ where:

$$S_i = \left\{\mathbf{x}_n \in \mathbb{R}^{D\times 1} : \mathbf{x}_n = \mathbf{A}_i \mathbf{z}_n^i \right\}_{n=1}^{N} \quad i \in \{1,\ldots,C\} \tag{1}$$

where C represents number of subspaces, which coincides with the number of clusters, $\{\mathbf{A}_i \in \mathbb{R}^{D\times d_i}\}_{i=1}^{C}$ represent bases of subspaces, $\{d_i \ll D\}_{i=1}^{C}$ represent dimensions of subspaces, and $\{\mathbf{z}_n^i \in \mathbb{R}^{d_i}\}_{i=1}^{C}$ stand for representations of data samples $\{\mathbf{x}_n\}_{n=1}^{N}$. In the most general formulation of SC problem, the goal is to identify the number of subspaces, the bases of these subspaces, their dimensions, and to cluster the data points according to the subspaces from which they are generated [5]. It is frequently assumed that the number of subspaces C is known *a priori*. Moreover, SC primarily focuses on clustering [5]. Under this specific data model, SC algorithms aim to learn the representation matrix $\mathbf{Z} \in \mathbb{R}^{N\times N}$, from which the data affinity matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is derived as follows:

$$\mathbf{W} = \frac{1}{2}\left(|\mathbf{Z}| + |\mathbf{Z}|^{\mathrm{T}}\right) \tag{2}$$

The diagonal degree matrix $\mathbf{D}$, based on the affinity matrix $\mathbf{W}$, is computed as:

$$d_{ii} = \sum_{j=1}^{N} w_{ij} \quad i \in \{1,..,N\}. \tag{3}$$

Using Eqs. (2) and (3), the normalized graph Laplacian matrix is computed as [4]:

$$\mathbf{L} = \mathbf{I} - (\mathbf{D})^{-1/2}\mathbf{W}(\mathbf{D})^{-1/2} \tag{4}$$

The spectral clustering algorithm is then applied to $\mathbf{L}$ to assign cluster labels to the data points: $\mathbf{F} \in \mathbb{N}_0^{N\times C}$. Based on the self-expressive data model $\mathbf{X}$=$\mathbf{XZ}$, many SC algorithms focus on learning the representation matrix $\mathbf{Z}$ by solving the following optimization problem:

$$\min_{\mathbf{Z}} \frac{1}{2}\| \mathbf{X} - \mathbf{XZ} \|_F^2 + \lambda f(\mathbf{Z}) + \tau g(\mathbf{Z}) \quad \text{s.t.} \quad \mathrm{diag}(\mathbf{Z}) = \mathbf{0}. \tag{5}$$

In Eq. (5), f and g are regularization functions imposed on $\mathbf{Z}$, while λ and τ are regularization constants (hypeparameters). For sparse SC (SSC) [5], $f(\mathbf{Z}) = \| \mathbf{Z} \|_1$ and τ=0. For LSR SC [7], $f(\mathbf{Z}) = \| \mathbf{Z} \|_F$ and τ=0. For S0L0 low-rank sparse SC (LRSSC) [8], $f(\mathbf{Z}) = \| \mathbf{Z} \|_{s_0}$ and $g(\mathbf{Z}) = \| \mathbf{Z} \|_0$. If the constraint diag($\mathbf{Z}$)=$\mathbf{0}$ is removed from Eq. (5), the LSR SC algorithm has an analytical solution:

$$\mathbf{Z} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}. \tag{6}$$

For LRR SC, SSC, and S0L0 LRSSC solutions of Eq. (5) are obtained iteratively using the alternating direction method of multipliers (ADMM) [25]. It is straightforward to derive the nonlinear kernel-based version of the LSR SC algorithm (6) by replacing $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ with the Gramm matrix $\mathbf{K}(\mathbf{X},\mathbf{X})$:

$$\mathbf{Z} = (\mathbf{K}(\mathbf{X},\mathbf{X}) + \lambda\mathbf{I})^{-1}\mathbf{K}(\mathbf{X},\mathbf{X}). \tag{7}$$

Thus, the kernel version of the LSR SC also depends on one hyperparameter, λ. However, the number of hyperparameters increases depending on the choice of the kernel function. For instance, if we select the Gaussian kernel $\kappa(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\| \mathbf{x}_i - \mathbf{x}_j\| /2\sigma^2\right)$, the variance σ^2 becomes an additional hyperparameter. Graph filtering can be an alternative to kernel methods for clustering data generated from manifolds [12]. This method generates a smoothed (filtered) version of the feature matrix as follows:

$$\overline{\mathbf{X}}^{\mathrm{T}} = \left(\mathbf{I} - \frac{\mathbf{L}}{2}\right)^{k}\mathbf{X}^{\mathrm{T}} \tag{8}$$

where k is a non-negative integer. Instead of applying the selected SC algorithm to the original data matrix $\mathbf{X}$, it is applied to the graph-filtered data $\overline{\mathbf{X}}$. The data adjacency matrix $\mathbf{W}$ is typically computed from Eq. (2), where $\mathbf{Z}$ is the representation matrix estimated by a SC using self-representation model $\mathbf{X}$=$\mathbf{XZ}$. Since the goal is to apply SC algorithm to $\overline{\mathbf{X}}$, we implement a graph-filtering version of the algorithm in an iterative manner [12]. This process is illustrated in Algorithm 1, which combines graph filtering with the LSR SC. Unlike kernel methods, it does not suffer from problems analogous to kernel selection.

In many real-world experiments, collected data originate from multiple modalities. For example, the same documents may be available in multiple languages [26], while different descriptors can be generated for the same image [27]. These scenarios lead to the multi-view SC (MVSC) problem [8,9]. A multi-view dataset composed of V views is denoted as $\left\{\mathbf{X}^{(v)} \in \mathbb{R}^{D_v\times N}\right\}_{v=1}^{V}$. The objective of MVSC algorithms is to learn a data affinity matrix that integrates all the views and then apply spectral clustering to assign labels to data points. The LMVSC method proposed in [10] has linear complexity in terms of the number of samples and solves the following optimization problem:

$$\min_{\mathbf{Z}^{(v)}} \sum_{v=1}^{V} \| \mathbf{X}^{(v)} - \mathbf{A}^{(v)}(\mathbf{Z}^{(v)})^{\mathrm{T}} \|_F^2 + \lambda \| \mathbf{Z}^{(v)} \|_F^2 \ \text{s.t.}\ \mathbf{0} \le \mathbf{Z}^{(v)},\ (\mathbf{Z}^{(v)})^{\mathrm{T}}\mathbf{1} = \mathbf{1}. \tag{9}$$

Algorithm 1
Graph filtering least squares subspace clustering.

Inputs: Feature (data) matrix $\mathbf{X}$, stopping criterion ε=10^{-4}.
Parameters: Filter order k, λ-regularization constant
1: Initialize t=0 and $\overline{\mathbf{X}}_1 = \mathbf{X}$
2: **repeat**
3: $t \leftarrow t+1$
4: $\mathbf{Z}_t \leftarrow \left(\overline{\mathbf{X}}_t^{\mathrm{T}}\overline{\mathbf{X}}_T + \lambda\mathbf{I}\right)^{-1}\overline{\mathbf{X}}_t^{\mathrm{T}}\overline{\mathbf{X}}_t$
5: $\mathbf{W}_t = \frac{1}{2}\left(|\mathbf{Z}_t| + |\mathbf{Z}_t|^{\mathrm{T}}\right)$
6: $\mathbf{L}_t = \mathbf{I} - \mathbf{D}_t^{-1/2}\mathbf{W}_t\mathbf{D}_t^{-1/2}$
7: $\overline{\mathbf{X}}_{t+1}^{\mathrm{T}} = \left(\mathbf{I} - \frac{\mathbf{L}_t}{2}\right)^{k}\mathbf{X}^{\mathrm{T}}$
8: **until** $\| \mathbf{W}_t - \mathbf{W}_{t-1} \|_F^2 \le \varepsilon$
Output: Graph-filtered adjacency matrix $\mathbf{W} \leftarrow \mathbf{W}_t$

In Eq. (9), $\mathbf{A}^{(v)} \in \mathbb{R}^{D_v \times M}$ represents the M anchors for the v-th view, $\mathbf{Z}^{(v)} \in \mathbb{R}^{N \times M}$ is the representation matrix for the v-th view, and $\mathbf{1}$ is a vector of ones. The view-dependent data adjacency matrices are constructed as:

$$\mathbf{W}^{(v)} = \widehat{\mathbf{Z}}^{(v)}(\widehat{\mathbf{Z}}^{(v)})^{\mathrm{T}}\ \widehat{\mathbf{Z}}^{(v)} = \mathbf{Z}^{(v)}\boldsymbol{\Sigma}^{-1/2},\ \boldsymbol{\Sigma}_{ii} = \sum_{j=1}^{N}\mathbf{Z}_{ji}^{(v)}. \tag{10}$$

The final data adjacency matrix that represents all views is obtained as:

$$\mathbf{W} = \frac{1}{V}\sum_{v=1}^{V}\mathbf{W}^{(v)}. \tag{11}$$

The outlined LMVCS algorithm has two hyperpamerters: M and λ.

Almost all (traditional) SC algorithms follow a two-stage process to assign cluster labels to data points. In the first stage, spectral decomposition of data affinity matrix is performed, resulting in a continuous relaxation of the binary cluster indicator matrix. In the second stage, the k-means algorithm is used to round this continuous relaxation into a binary indicator matrix. However, this two-stage approach can lead to severe information loss and performance degradation [28]. To address this issue, the multi-view clustering via multiple Laplacian embeddings (MCMLE) method was proposed in [28]. It aims to learn a unified $N \times C$ binary indicator clustering matrix $\mathbf{Y} \in Ind$ that represents all views. This matrix is obtained by solving the following optimization problem:

$$\begin{aligned}&\min_{\mathbf{F}^{(1)},\ldots,\mathbf{F}^{(V)},\mathbf{Y},\boldsymbol{\lambda}} \sum_{v=1}^{V}\lambda^{(v)}\left(tr\left(\mathbf{F}^{(v)^{\mathrm{T}}}\mathbf{L}^{(v)}\mathbf{F}^{(v)}\right) - \alpha tr\left(\mathbf{F}^{(v)\mathrm{T}}\mathbf{D}^{(v)\frac{1}{2}}\mathbf{Y}\right)\right) + \beta\|\boldsymbol{\lambda}\|_2^2,\\ &\text{s.t.}\ \mathbf{F}^{(v)\mathrm{T}}\mathbf{F}^{(v)} = \mathbf{I},\ \mathbf{Y} \in Ind,\ \sum_{v=1}^{V}\lambda^{(v)} = 1,\ \lambda^{(v)} \geq 0\end{aligned} \tag{12}$$

where $\alpha>0$ and $\beta>0$ are penalty parameters, $\{\mathbf{L}^{(v)}\}_{v=1}^{V}$ are view-specific normalized Laplacians in the form of Eq. (4), $\{\mathbf{F}^{(v)} \in \mathbb{R}^{N\times C}\}_{v=1}^{V}$ are view-specific partitions, and components of $\boldsymbol{\lambda} \in \mathbb{R}_{0+}^{V\times 1}$ are view-specific weights. The MCLME algorithm introduces two hyperparameters: α and β.

3. The proposed method

In this section, we present our pseudo-label approach to HPO for the selected SC algorithm. For the sake of readability, we first describe methodology for SC algorithms that depend on a single hyperparameter. Examples of such algorithms include LSR SC (6), sparse SC [5], low-rank representation SC [6], LMVSC (9)-(11) [9]. We then extend our approach to SC algorithms that depend on two hyperparameters. Examples in this category include kernel LSR SC (7), assuming a Gaussian kernel, S0L0 LRSSC algorithm [6], graph filtering LSR SC [12] also presented in Algorithm 1, and the MCLME algorithm [28].

3.1. Label-free self-guided hyperparameter optimization

Let us assume that the selected SC algorithm depends on a single hyperparameter λ, and that the predefined search space $\boldsymbol{\lambda} := [\lambda_1,\ldots,\lambda_M]$, $\boldsymbol{\lambda}\geq\mathbf{0}$, contains the optimal value λ^*. We also assume the hyperparameter values are ordered such that $\lambda_1<\lambda_2<\ldots<\lambda_M$. We use widely known clustering performance metric accuracy (ACC) and normalized mutual information (NMI) to measure alignment between pseudo-labels generated by proposed method at selected hyperparameter values. Let the cluster labels (pseudo-labels) generated by the SC algorithm for hyperparameter values $\boldsymbol{\lambda}$ be denoted by:

$$\{\mathbf{y}(\lambda_i) \in Ind := \{y_n(\lambda_i)\}_{n=1}^{N}\}_{i=1}^{M} \tag{13}$$

To simplify notation going forward, we shall use the shorthand $\mathbf{y}_i = \mathbf{y}(\lambda_i)$ and $\mathbf{y}_{i+1} = \mathbf{y}(\lambda_{i+1})$. Let $h(\mathbf{y}_i,\mathbf{y}_{i+1})$ represent either $ACC(\mathbf{y}_i,\mathbf{y}_{i+1})$ or $NMI(\mathbf{y}_i,\mathbf{y}_{i+1})$. By treating λ_{i+1} as a constant, we assume that h is either a monotonically increasing function of λ_i and λ_{i+2}:

$$h(\mathbf{y}_i,\mathbf{y}_{i+1}) \leq h(\mathbf{y}_{i+1},\mathbf{y}_{i+2}) \tag{14}$$

or a monotonically decreasing function of λ_i and λ_{i+2}:

$$h(\mathbf{y}_i,\mathbf{y}_{i+1}) \geq h(\mathbf{y}_{i+1},\mathbf{y}_{i+2}). \tag{15}$$

Next, we now locate the subinterval $[\lambda_i,\lambda_{i+1}]$ where $h(\mathbf{y}_i,\mathbf{y}_{i+1})$ is maximal, i.e.:

$$i = \underset{j}{\text{argmax}}\, h\left(\mathbf{y}_j,\mathbf{y}_{j+1}\right)\ 1 \leq j < M. \tag{16}$$

Let the iteration index be set to t=1, and denote $\lambda_1^{(t)} = \lambda_i$ and $\lambda_4^{(t)} = \lambda_{i+1}$. We now define hyperparameter search space at iteration t as:

$$\boldsymbol{\lambda}^{(t)} := \left[\lambda_1^{(t)}\ \lambda_2^{(t)}\ \lambda_3^{(t)}\ \lambda_4^{(t)}\right] \tag{17}$$

where $\lambda_2^{(t)} = \left(2\times\lambda_1^{(t)} + \lambda_4^{(t)}\right)/3$ and $\lambda_3^{(t)} = \left(\lambda_1^{(t)} + 2\times\lambda_4^{(t)}\right)/3$. We then use the selected SC algorithm to generate pseudo-labels $\mathbf{y}_2^{(t)}$ and $\mathbf{y}_3^{(t)}$, and estimate $h(\mathbf{y}_1^{(t)},\mathbf{y}_2^{(t)})$, $h(\mathbf{y}_2^{(t)},\mathbf{y}_3^{(t)})$ and $h(\mathbf{y}_3^{(t)},\mathbf{y}_4^{(t)})$. The subinterval $[\lambda_1^{(t+1)}, \lambda_4^{(t+1)}]$ is refined according to the following rules:

$$\begin{aligned}&\text{if } h(\mathbf{y}_1^{(t)},\mathbf{y}_2^{(t)}) \geq h(\mathbf{y}_2^{(t)},\mathbf{y}_3^{(t)}) \text{ and } h(\mathbf{y}_1^{(t)},\mathbf{y}_2^{(t)}) \geq h(\mathbf{y}_3^{(t)},\mathbf{y}_4^{(t)})\\ &\qquad \lambda_1^{(t+1)} \leftarrow \lambda_1^{(t)},\ \lambda_4^{(t+1)} \leftarrow \lambda_2^{(t)}\\ &\text{elseif } h(\mathbf{y}_2^{(t)},\mathbf{y}_3^{(t)}) \geq h(\mathbf{y}_1^{(t)},\mathbf{y}_2^{(t)}) \text{ and } h(\mathbf{y}_2^{(t)},\mathbf{y}_3^{(t)}) \geq h(\mathbf{y}_3^{(t)},\mathbf{y}_4^{(t)})\\ &\qquad \lambda_1^{(t+1)} \leftarrow \lambda_2^{(t)},\ \lambda_4^{(t+1)} \leftarrow \lambda_3^{(t)}\\ &\text{else}\\ &\qquad \lambda_1^{(t+1)} \leftarrow \lambda_3^{(t)},\ \lambda_4^{(t+1)} \leftarrow \lambda_4^{(t)}\\ &\text{end}\end{aligned} \tag{18}$$

We then increment the iteration index to $t\leftarrow t+1$ and repeat the process. HPO stops when the relative error criterion is satisfied:

$$\frac{\lambda_4^{(t+1)} - \lambda_1^{(t+1)}}{\lambda_1^{(t)}} \leq \varepsilon \tag{19}$$

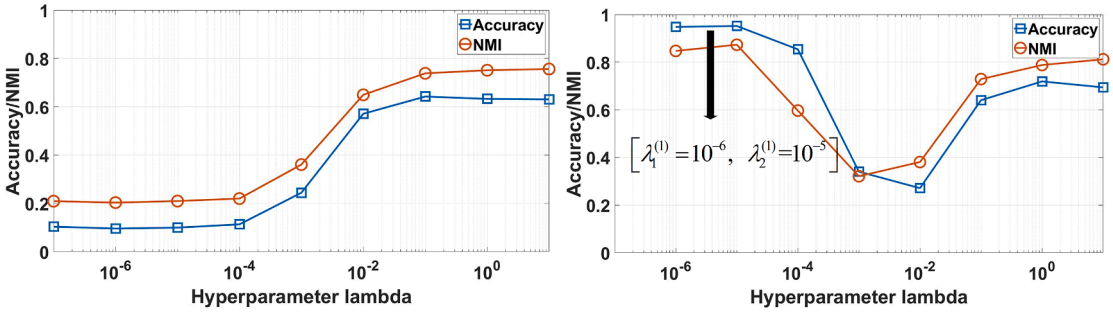
where ε is a predefined constant. In our experiments, reported in section 4, we set $\varepsilon = 0.001$. After the stopping criterion is met, the optimal hypeparameter value is obatined as:

$$\lambda^* = \frac{\lambda_4^{(t+1)} + \lambda_1^{(t+1)}}{2} \tag{20}$$

The experiments presented in section 4 brought to light an important issue related to the selection of the hyperparameter space $\boldsymbol{\lambda} := [\lambda_1,\ldots,\lambda_M]$. If two neighboring hyperparameters λ_i and λ_{i+1} are set "too close" to each other, the corresponding metric $h(\mathbf{y}_i,\mathbf{y}_{i+1})$ may exhibit a high value. However, the actual performance based on the true labels, $h(\mathbf{y}_i,\mathbf{y}^*)$, can be quite poor. This scenario is illustrated in Fig. 2, where both ACC and NMI, as functions of $\boldsymbol{\lambda} := [\lambda_1,\ldots,\lambda_M]$, are estimated by the oracle version of the LSR SC algorithm (left) and pseudo-labels based version of the LSR SC algorithm (right). We summarize our approach to HPO in SC in Algorithm 2.

3.2. Extension of proposed approach to HPO for SC algorithm with two hyperparameters

We now extend our approach to HPO for SC algorithms with two hyperparameters, such as kernel LSR SC (7) (assuming Gaussian kernel), S0L0 LRSSC algorithm [6], graph filtering LSR SC [12] (also presented in Algorithm 1), and the MCLME algorithm [28]. Let us assume that the hyperparameter space is defined by $\boldsymbol{\lambda} := \{\lambda_1,\ldots,\lambda_M\}$ and $\boldsymbol{\tau} := \{\tau_1,\ldots,\tau_L\}$. First, we preset the hyperparameter τ to a value $\tau = \tau_{L/2}$ (other options may also be considered). We then apply Algorithm 1 with the predefined



(a) $\mathrm{ACC}(\lambda_3=10^{-5})=0.1$ $\mathrm{NMI}(\lambda_3=10^{-5})=0.221$
$\mathrm{maxACC}(\lambda_7=10^{-1})=0.64$ $\mathrm{maxNMI}(\lambda_9=10)=0.756$

(b) $\max \mathrm{ACC}(\lambda_2=10^{-6},\lambda_3=10^{-5})=0.952$
$\max \mathrm{NMI}(\lambda_2=10^{-6},\lambda_3=10^{-5})=0.873$

Fig. 2. Illustration of possible problem in proposed approach to label-free HPO for LSR SC algorithm [7], on COIL20 dataset. Hyperparameter grid values $\lambda_1 = 10^{-7}$ and $\lambda_2 = 10^{-6}$ are set too close. (**a**) Accuracy and NMI at $\lambda := \{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$. The value 0.6519 of maximal accuracy computed with external (hard) labels occurs at $\lambda_7 = 0.1$. As for NMI, the maximal value 0.7733 occurs at $\lambda_9 = 10$. (**b**) Accuracy and NMI computed between pseudo-labels generated by the same grid values of λ as previously. Maximal values occur between pseudo-labels generated by $\lambda_2 = 10^{-6}$ and $\lambda_3 = 10^{-5}$. Consequently, new iteration 1 begins on the interval with borders $\lambda_1^{(1)} = \lambda_2 = 10^{-6}$ and $\lambda_2^{(1)} = \lambda_3 = 10^{-5}$. That is incorrect.

hyperparameter space $\boldsymbol{\lambda}$ to compute λ^*. Afterward, we fix λ at the obtained value $\lambda = \lambda^*$, and apply Algorithm 1 with the predefined hyperparameter space $\boldsymbol{\tau}$ to compute τ^*. Described approach is based on implicit assumptions on $h_{\tau_{L/2}}(\mathbf{y}_i, \mathbf{y}_{i+1})$ and $h_{\lambda^*}(\mathbf{y}_i, \mathbf{y}_{i+1})$, analogous to Eqs. (16) and (17). Specifically, we assume that $h_{\tau_{L/2}}(\mathbf{y}_i, \mathbf{y}_{i+1})$ is a monotonic function of τ_i and τ_{i+1}, and $h_{\lambda^*}(\mathbf{y}_i, \mathbf{y}_{i+1})$ is a monotonic function of λ_i and λ_{i+1}. The subscripts $\tau_{L/2}$ and superscript λ^* indicate that the performance indices are computed with these parameters. Described method for HPO of the SC algorithms with two hypeparameters can be straightforwardly extended to HPO of the SC algorithms that depend on three or more hyperparameters.

3.3. Extension of proposed approach to HPO of SC algorithm for out-of-sample data

Many SC algorithms are unable to cluster out-of-sample (also known as unseen or test) data. In other words, to cluster an unseen data point, the algorithm must be re-run again on the entire data, including the new data point. This requirement significantly reduces the scalability and applicability of these algorithms in large-scale or online clustering problems. Here, we address the problem of clustering out-of-sample data by LFSG SC algorithm by formulating it as a minimization problem based on the point-to-a-subspace distance criterion. Using the partitions obtained by the selected algorithm on the in-sample dataset, we estimate the subspace bases [13]:

$$\left\{\mathbf{X}_c \leftarrow \mathbf{X}_c - \left[\underbrace{\overline{\mathbf{x}}_c \ldots \overline{\mathbf{x}}_c}_{N_c \text{ times}}\right]\right\}_{c=1}^{C} \tag{21}$$

where $\overline{\mathbf{x}}_c = \frac{1}{N_c}\sum_{n=1}^{N_c}\mathbf{X}_c(n)$, $\bigcup_{c=1}^{C}\mathbf{X}_c = \mathbf{X}$, $\sum_{c=1}^{C}N_c = N$, and C is the number of clusters. From $\{\mathbf{X}_c = \mathbf{U}_c\boldsymbol{\Sigma}_c(\mathbf{V}_c)^{\mathrm{T}}\}_{c=1}^{C}$ we estimate the orthonormal bases by extracting the first d left singular vectors of each partition, i.e. $\{\mathbf{U}_c \in \mathbb{R}^{D\times d}\}_{c=1}^{C}$ [13]. For a test data point $\mathbf{x}$, we compute the point-to-a-subspace distances $\{d_c = \| \widetilde{\mathbf{x}}_c - \mathbf{U}_c(\mathbf{U}_c)^{\mathrm{T}}\widetilde{\mathbf{x}}_c\|_2\}$ as follows:

$$c = \underset{c\in\{1,\ldots,C\}}{\operatorname{argmin}} d_c \tag{22}$$

where $\widetilde{\mathbf{x}}_c = \mathbf{x} - \overline{\mathbf{x}}_c$. We assign the label $\{c\}_{c=1}^{C}$ to the test data point:

$$[\pi(\mathbf{x})]_c = \begin{cases} 1, & \text{if } c = \underset{c\in\{1,\ldots,C\}}{\operatorname{argmin}} d_c \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Algorithm 2
Proposed approach to a HPO for a single hyperparameter SC algorithm (e.g., LSR SC Eq. (6), [7]).

Input: Feature (data) matrix $\mathbf{X}$, initial hyperparameter space $\lambda := [\lambda_1, \ldots, \lambda_M]$, stopping criterion $\varepsilon = 0.001$.
1: Locate the subinterval $[\lambda_i, \lambda_{i+1}]$ where $h(\mathbf{y}_i, \mathbf{y}_{i+1})$ is maximal, i.e.:
$i = \underset{j}{\operatorname{argmax}} h(\mathbf{y}_j, \mathbf{y}_{j+1})$ $1 \le j < M$ (18).
2: Set t=1, and denote $\lambda_1^{(t)} = \lambda_i$, $\lambda_4^{(t)} = \lambda_{i+1}$, $\mathbf{y}_1^{(t)} = \mathbf{y}_i$ and $\mathbf{y}_4^{(t)} = \mathbf{y}_{i+1}$.
3: Set $\lambda_2^{(t)} = (2\times\lambda_1^{(t)} + \lambda_4^{(t)})/3$ and $\lambda_3^{(t)} = (\lambda_1^{(t)} + 2\times\lambda_4^{(t)})/3$.
4: Use the selected SC algorithm to generate pseudo labels $\mathbf{y}_2^{(t)}$ and $\mathbf{y}_3^{(t)}$.
5: Estimate $h(\mathbf{y}_1^{(t)}, \mathbf{y}_2^{(t)})$, $h(\mathbf{y}_2^{(t)}, \mathbf{y}_3^{(t)})$ and $h(\mathbf{y}_3^{(t)}, \mathbf{y}_4^{(t)})$.
6: Use Eq. (18) to select borders of the next subinterval $\lambda_1^{(t+1)}$ and $\lambda_4^{(t+1)}$.
7: If the relative error criterion (13) is met, proceed to step 13.
8: Compute $\lambda_2^{(t+1)} = (2\times\lambda_1^{(t+1)} + \lambda_4^{t+1})/3$, and $\lambda_3^{(t+1)} = (\lambda_1^{(t+1)} + 2\times\lambda_4^{t+1})/3$.
9: Use selected SC algorithm to generate pseudo-labels $\mathbf{y}_1^{(t+1)}$, $\mathbf{y}_2^{(t+1)}$, $\mathbf{y}_3^{(t+1)}$ and $\mathbf{y}_4^{(t+1)}$.
10: Estimate $h(\mathbf{y}_1^{(t+1)}, \mathbf{y}_2^{(t+1)})$, $h(\mathbf{y}_2^{(t+1)}, \mathbf{y}_3^{(t+1)})$ and $h(\mathbf{y}_3^{(t+1)}, \mathbf{y}_4^{(t+1)})$.
11: Increment $t \leftarrow t+1$.
12: Return to step 6.
13: Compute optimal hyperparameter value λ* using Eq. (20).
Output: The optimal hyperparameter value λ^*.

3.4. Formulation of proposed approach to HPO for out-of-sample extension of the kernel LSR SC algorithm

To apply the point-to-a-subspace distance criterion for clustering out-of-sample data in reproducible kernel Hilbert space (RKHS), the approach presented in section 3.3 has to be slightly adapted. Due to the space limitation, we provide full derivation of proposed approach to HPO for out-of-sample extension of the kernel LSR SC algorithm in the supplement.

3.5. Interpretable LFSG SC

As with many other complex prediction models, SC algorithms generally suffer from the issue of interpretability. In our approach to LFSG SC, we propose a method to interpret or explain the results obtained by the LFSG SC algorithm. Notably, this proposed approach is also applicable to the oracle versions of the corresponding SC algorithm. Our method relies on the visualization of subspace bases estimated from the in-sample data, using the clustering partitions produced by the selected SC algorithm. Let the SVD of each in-sample cluster partition be represented as:

$$\left\{\mathbf{X}_c = \mathbf{U}_c\mathbf{\Sigma}_c(\mathbf{V}_c)^{\mathrm{T}}\right\}_{c=1}^{C}. \tag{24}$$

We aim to visualize each partition. To achieve this, we compute the representative of each cluster as:

$$\left\{\mathbf{a}_c \in \mathbb{R}^{D\times 1} = \sum_{j=1}^{d}\mathbf{U}_c(:,j)\mathbf{\Sigma}_c(j,j)\right\}_{c=1}^{C} \tag{25}$$

In Eq. (25), d represents the subspace dimension, which is known *a priori*. For example, face images of each subject in the EYaleB dataset lie approximately in a subspace of $d = 9$ dimensions [29], while handwritten digits such as those in the MNIST dataset, appriximately lie in a subspace of $d = 12$ dimensions [30]. Since SC algorithms typically vectorize image data samples, we need to matricize the representative vectors to recover their original 2D form. Let us assume the dimensions of the imaging data are D_x and D_y, where $D{=}D_x \times D_y$. Using MATLAB notation, we reconstruct the images of the cluster representatives as:

$$\left\{\mathbf{A}_c \in \mathbb{R}^{D_x\times D_y} = \mathit{reshape}\left(\mathbf{a}_c, D_x, D_y\right)\right\}_{c=1}^{C}. \tag{26}$$

We demonstrate the proposed visualization method on the USPS dataset [31], which contains images of handwritten numerals organized into $C{=}10$ clusters. Each grayscale image has dimensions $D_x = D_y = 16$. Fig. 3 shows the visualization results for correctly labeled data, as well as for data labeled by both the oracle and LFSG versions of the S0L0 LRSSC algorithm [6]. Despite the inherent rotational indeterminacy of the proposed visualization, the numerals corresponding to each group are recognizable, allowing us to explain the decisions of both the oracle LFGS versions of the S0L0 LRSSC algorithm. The LFSGSC method can, at a conceptual level, be compared with the LIME algorithm [32]. LIME's core idea is to perturb the features and track any changes in the model's prediction. This helps to identify which features significantly influence the model's decision-making process. Similarly, LFSGSC generates pseudolabels and visualize the subspace using the resulting clustering partitions. This allows users to assess importance and influence of specific combination of hyperparameters.

4. Experiments

4.1. Single-view SC algorithms

We evaluate the proposed approach to HPO on LSR SC algorithm (6), [7], graph-filtering version of LSR SC algorithm in Algorithm 1, kernel LSR SC algorithm (7), [7], sparse SC (SSC) algorithm [5], and S0L0 low rank sparse SC (LRSSC) algorithm [6]. For this evaluation we use six datasets: MNIST [33], USPS [31], EYaleB [34], ORL [35], COIL20, and COIL100 [36]. For each algorithm, we compute accuracy and normalized mutual information (NMI) to select the optimal hyperparameter value over the specified search space. This is done using both the true labels (the oracle) and pseudo-labels, following the label-free self-guided (LFSG) approach. For clarity, we define the oracle version of a given algorithm as the clustering performance achieved when its hyperparameters are optimized using the true ground-truth labels. After determining the optimal hypeparameter value, we compute accuracy, NMI and F1-score using the true labels to assess clustering performance. To evaluate the robustness of the algorithms, we randomly split each dataset into in-sample and out-of-sample partitions and repeat the clustering process 25 times. This allows us to statistically compare the oracle and LFSG versions of each selected SC algorithm using the Wilcoxon rank-sum test implemented by MATLAB command ranksum. The null hypothesis of the test assumes that the data come from continuous distributions with equal medians, at a 5 % significance level. A p-value greater than 0.05 indicates acception of the null hypothesis. The main information on datasets used in the experiments are summarized in Table 1. Due to space limitation we present in the main paper results after HPO, using the accuracy metric in a form of bar diagrams. Full numerical results, including information about statistical significance, for both accuracy and NMI metrics are presented in the supplement.

4.1.1. Least squares regression SC algorithm

Fig. 4 presents the clustering performance of the oracle and LFSG versions of the LSR SC algorithm for hyperparameter selection based on ACC criteria. Table S1 in supplement gives complete numerical results for hyperparameter selection based on ACC and NMI criteria including statistical significance analysis. The p-values are provided at the 95 % confidence interval for instances when the statistical difference between the oracle and LFSG versions was insignificant, i.e., when the null hypothesis (equal medians) is accepted. These findings suggest that the proposed LFSG LSR SC algorithm performs comparably to its oracle counterpart. For further details on the hyperparameter λ search space for each dataset, we direct interested readers to the publicly available code. As shown in Fig. 4, the LFSG version on in-sample data typically has up to 4 % lower clustering performance to the oracle version, while on out-of-sample data, the performance difference is typically less than 1 %. This suggests that the partitions generated by the LFSG version on in-sample data are sufficiently robust to estimate accurate subspace bases for clustering out-of-sample (test) data.

4.1.2. Kernel least squares regression SC algorithm

For the kernel LSR SC algorithm (7), we selected the Gaussian kernel, introducing variance σ^2 as an additional hyperparameter alongside λ. Fig. 5 presents the clustering performance of the oracle and LFSG versions of the kernel LSR SC algorithm for hyperparameter selection based on ACC criteria. Table S2 in supplement gives complete numerical results for hyperparameter selection based on ACC and NMI criteria including statistical significance analysis. As observed in Table S2, the oracle version typically outperforms the LFSG version by 3-4 %. However, in several cases, the performance difference on in-sample data is statistically insignificant. Notably, for out-of-sample data, the LFSG

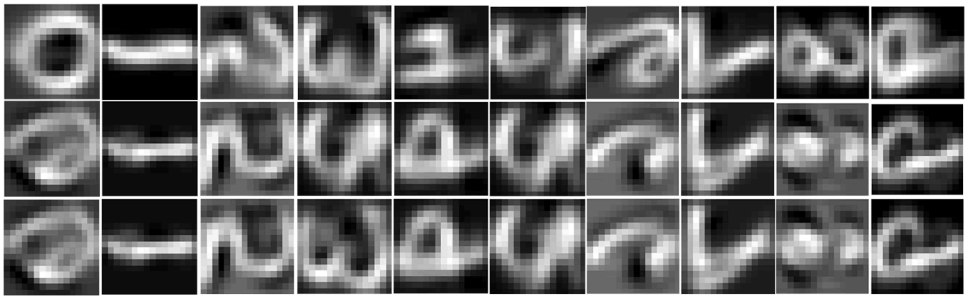


Fig. 3. Visualization of subspace bases representing the numerals in the USPS dataset [31], from 0 to 9 in order. The bases are estimated from 50 randomly selected data samples per cluster (numeral). Clustering is performed using the S0L0 LRSSC algorithm [6]. From top to bottom: partitions are composed of (1) correctly labeled data, (2) data labeled by the oracle version of the S0L0 LRSSC algorithm, (3) data labeled by the LFSG version of the S0L0 LRSSC algorithm. The clustering accuracy of both versions is approximately 85 %.

Table 1
Main information on datasets used in the experiments with single-view SC algorithms.

Dataset	#Sample	#Feature	#Cluster	#in-sample/out-of-sample per group
MNIST	10000	28 × 28	10	50/50
USPS	7291	16 × 16	10	50/50
EYaleB	2432	48 × 42	38	43/21
ORL	400	32 × 32	40	7/3
COIL20	1440	32 × 32	20	26/26
COIL100	7200	32 × 32	100	26/26

version achieves significantly better performance on all datasets. This suggests that the partitions generated by the LFSG version on in-sample data are more robust for estimating accurate subspace bases for clustering out-of-sample (test) data. It can be argued that the oracle version could also produce more robust partitions, if a denser hyperparameter search space were employed. However, this would significantly increase computational complexity.

4.1.3. Graph filtering least squares regression SC algorithm

As shown in Algorithm 1, the graph-filtering LSR SC algorithm introduces an additional hyperparameter, the filter order k, alongside the regularization constant λ. Unlike the LSR SC and kernel LSR SC algorithms, we have not yet formulated an out-of-sample extension for the graph-filtering LSR SC algorithm. Therefore, all results presented in Fig. 6 and Table S3 in the supplement are based on in-sample-data. As seen in Table S3, the performance difference between the oracle and LFSG versions is typically less than 4 %, and sometimes below 2 %. For the ORL, COIL20, and partially USPS datasets, there is no statistically significant difference in performance between the two versions. In several instances, there is also no statistically significant difference in hyperparameter estimates obtained by the oracle and LFSG versions.

4.1.4. Sparse SC algorithm

In the implementation of the sparse SC algorithm, we distinguish between two cases: when the error/noise term is Gaussian and when it is sparse, see [5]. The first case of the SSC algorithm is derived from Eq. (5) by setting $f(\mathbf{Z}) = \| \mathbf{Z} \|_1$ and τ=0, requiring the coefficient representation matrix $\mathbf{Z}$ to be sparse. The second case, robust to outliers, is defined by:

$$\min_{\mathbf{Z}} \| \mathbf{X} - \mathbf{X}\mathbf{Z} \|_1 + \lambda \| \mathbf{Z} \|_1 \text{ s.t. } \operatorname{diag}(\mathbf{Z}) = \mathbf{0}. \tag{27}$$

Eq. (27), represents robust version of the SSC algorithm, used to cluster the extended YaleB dataset. All other datasets were clustered using the non-robust version of the SSC algorithm. The SSC algorithm has one hyperparameter, λ. In its implementation, as outlined in [5], a scaled regularization constant is introduced $\alpha = \lambda / \mu$. For the non-robust version, $\mu \triangleq \min_{i} \max_{j \neq i} |\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j|$, $i, j = 1, N$, and for the robust version, $\mu \triangleq \min_{i} \max_{j \neq i} \| \mathbf{x}_j \|_1$, $i, j = 1, N$, giving us $\lambda = \alpha \times \mu$. When defining the search space, we preset $\alpha \in \mathbb{N}_+$. As seen in Fig. 7 and Table S4 in the supplement, for ORL, EYaleB, MNIST, USPS, and COIL20 the performance of the oracle version on in-sample data is mostly up to 6 % better than the LFSG version. For COIL100 dataset, the LFSG version yields up to 2 % better performance. On out-of-sample data, the difference in performance is in many cases statistically insignificant.

4.1.5. Low-rank sparse S_0/ℓ_0 SC algorithm

The S0L0 LRSSC algorithm [6] is derived from Eq. (5) by setting $f(\mathbf{Z}) = \| \mathbf{Z} \|_{S_0}$ and $g(\mathbf{Z}) = \| \mathbf{Z} \|_0$. In our implementation, as outlined in Algorithm 2 of [6], we further set λ+τ=1, meaning τ=1-λ. The second hyperparameter, α>0, is the penalty constant in the ADMM-based implementation of the S0L0 LRSSC algorithm. As shown in Fig. 8 and Table S5 in supplement, for the ORL dataset, the clustering performance of the LFSG version is up to 15 % lower than the oracle version, both on in-sample and out-of-sample data. On other datasets, the LFSG version's performance is typically up to 6 % lower than that of the oracle version, with similar results observed for out-of-sample data.

4.2. Multi-view SC algorithm

We also evaluated the performance of the proposed HPO approach on the multi-view LMVSC algorithm [9] and multi-view MLME algorithm [28]. As there are no out-of-sample extensions for the LMVSC and MLME algorithms, the results presented in Tables S7 and S8 pertain only to in-sample data. We compare clustering performance of MLME and LMVSC algorithms with fast parameter free multi-view subspace clustering algorithm with consensus anchor guidance (FPMVS-CAG), [10], on three multi-view datasets: Handwritten-numerals, BBC, and Caltech101-20. They are available at links provided in sections 4.2.1 and 4.2.2. Our motivation is to demonstrate that proposed LFSG version of MLME and LMVSC algorithms outperforms the parameter-free FPMVS-CAG which justifies application of proposed method to the existing SC algorithms. Performance metrics were calculated using in-sample data from 70 % randomly selected samples per cluster. Main information on used multi-view datasets are summarized in Table 2. Results obtained by FPMVS-CAG algorithm are given in Table 3.

4.2.1. Multi-view LMVSC algorithm

The LMVSC algorithm, formulated in Eq. (9), introduces two hyperparameters: the number of anchors M and the regularization constant α. The MATLAB code for the original version of this algorithm, provided by the authors, is available at https://github.com/sckangz/LMVSC. Fig. 9 and Table S7 in supplement report results on 25 runs, using data randomly selected from the 70 % of the total number of samples per cluster. As shown in [10], the optimal hyperparameter values for the Handwritten_numerals dataset are M=10 and α=10^{-3}. In comparison with [10], we slightly modified the search space, M∈{10, 15, 25, 50} and α∈{10^{-4} ,10^{-3}, 10^{-2}, 10^{-1}, 1, 10}, which improved the oracle's performance compared to the original results in [11]. As shown in Table S7, the oracle version estimated the optimal hyperparameter values M=25 and α=10^{-4} using either ACC or NMI as the selection

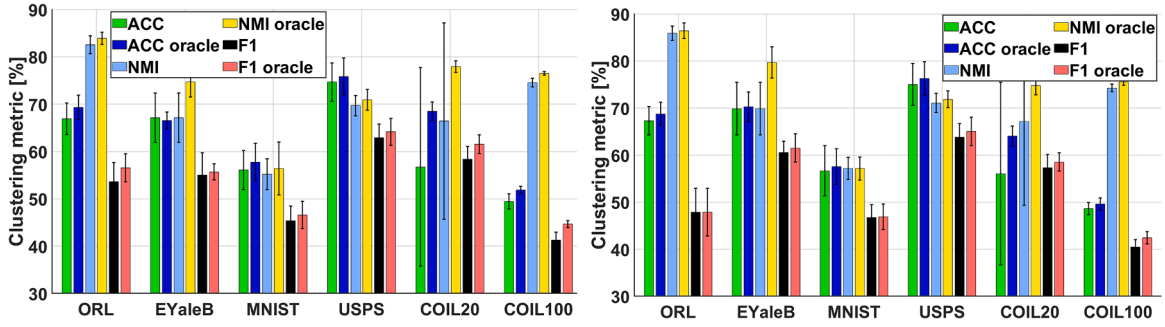


Fig. 4. The clustering performance of the oracle and LFSG versions of the LSR SC algorithm on the in-sample data (left) and out-of-sample data (right).

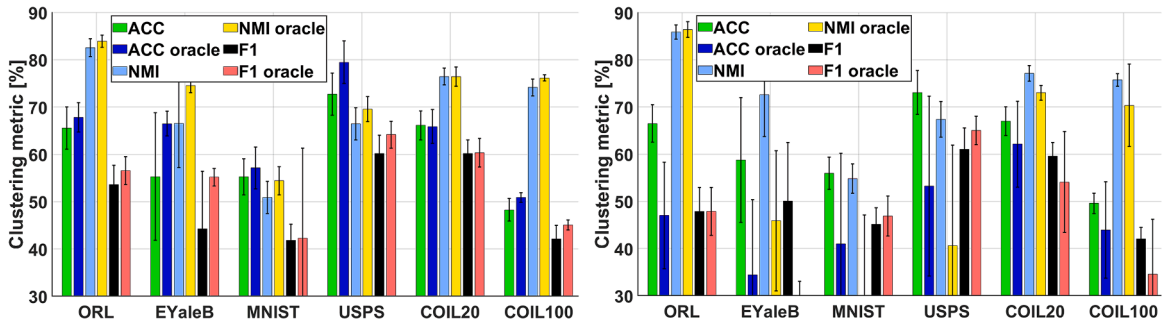


Fig. 5. The clustering performance of the oracle and LFSG versions of the kernel LSR SC algorithm on the in-sample data (left) and out-of-sample data (right).

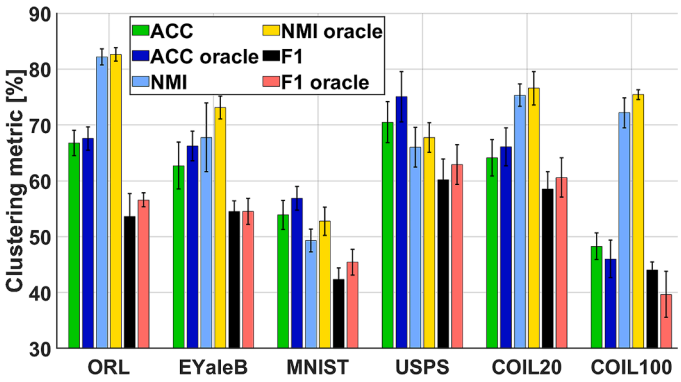


Fig. 6. The clustering performance of the oracle and LFSG versions of the graph filtering LSR SC algorithm.

metric. The LFSG version estimated hyperparameter values with a slight displacement relative to the oracle values. As can be seen in Table S7, the LFSG version's performance is typically up to 7 % lower than that of the oracle version. In comparison with parameter-free FPMVS-CAG algorithms, the LFSG LMVSC algorithm achieves better results on Handwritten numerals and BBC datasets, and worse results on Caltech101-20 dataset. Arguably, the last result could be improved by evaluating the quality of clustering partitions visualization but that would increase computational complexity of LFSG LMVSC method.

4.2.2. *Multi-view MLME algorithm*

The MLME algorithm, formulated in Eq. (12), introduces two hyperparameters: α and β. The MATLAB code for the original version of this algorithm, provided by the authors, is available at https://github.com/Ekin102003/MCMLE. We compared the oracle and LFSG versions using the BBC dataset and Handwritten numerals dataset. Fig. 10 and Table S8 in supplement report results after 25 runs using data randomly selected from the 70 % of the total number of samples per cluster. Researchers in [28] shown that the optimal values of the hyperparameters for the BBC dataset are $\beta \geq 10$ and $\alpha \in [0.0005, 0.01]$. As shown in Table S8, the oracle version correctly estimated the hyperparameters using either ACC or NMI as the selection metric. The LFSG version also estimated hyperparameter values within the suggested range. The same observation applies to results obtained on the 70 % randomly sampled data, although the mean values deviated from those values obtained on the full dataset.

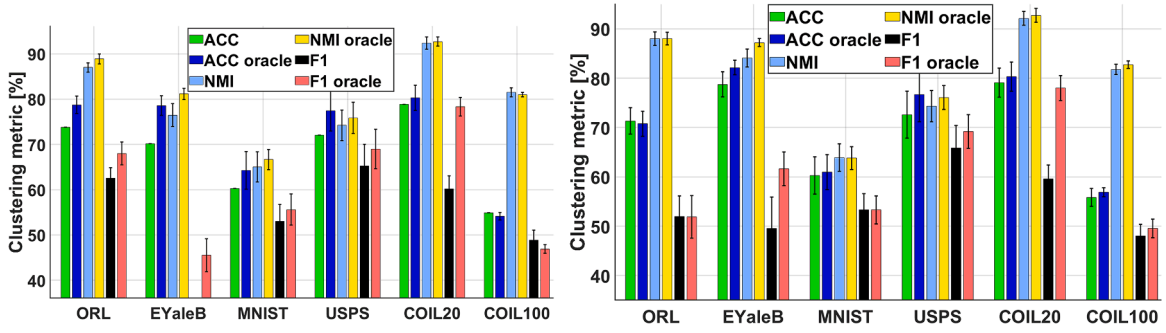


Fig. 7. The clustering performance of the oracle and LFSG versions of the SSC algorithm on the in-sample (left) and out-of-sample (right) data.

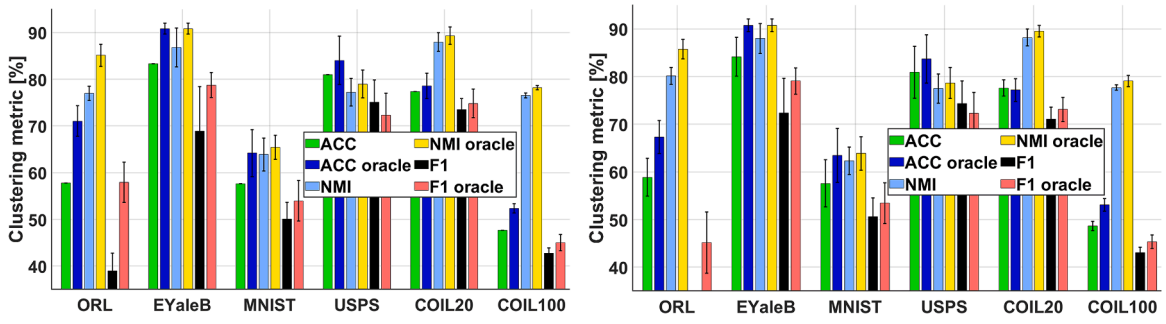


Fig. 8. The clustering performance of the oracle and LFSG versions of the S0L0 LRSSC algorithm on the in-sample (left) and out-of-sample (right) data.

Table 2
Main information on datasets used in the experiments with multi-view SC algorithms.

Dataset	#Sample	#Views	#Cluster
Handwritten numerals	2000	6	10
BBC	685	4	5
Caltech101-20	2386	6	20

Table 3
The clustering performance of the FPMVS-CAG algorithm. Performance metrics were calculated using in-sample data from 70 % randomly selected samples per cluster.

Dataset	ACC [%]	NMI [%]	F1 score [%]
Handwritten_numerals	82.25 ± 0	79.30 ± 0	75.60 ± 0
BBC	32.26 ± 0	2.97 ± 0	27.59 ± 0
Caltech101-20	65.47 ± 0	63.26 ± 0	69.05 ± 0

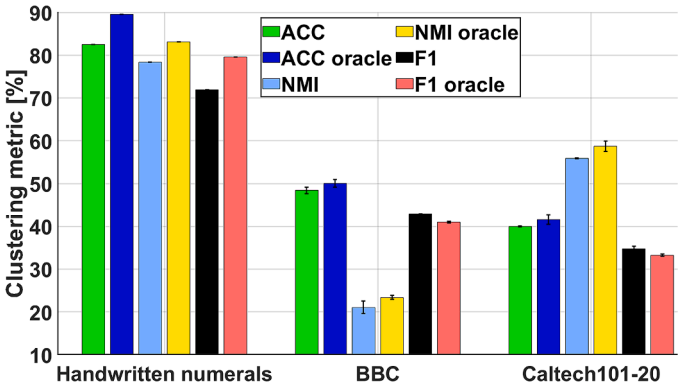


Fig. 9. The clustering performance of the oracle and LFSG versions of the LMVSC algorithm.

4.2.3. Fast parameter-free multi-view subspace clustering with consensus anchor guidance

Researchers in [9] proposed the fast parameter-free multi-view subspace clustering with consensus anchor guidance (FPMVS-CAG) method, with the MATAB code available at: https://github.com/wangsiwei2010/FPMVS-CAG. In this algorithm anchor selection and subspace graph construction are conducted into a unified optimization framework. Importantly, the algorithm is proven to have linear time complexity with respect to the number of samples. Even more important, the algorithm can learn the anchor-graph structure without hyperparameters.

5. Conclusion

Performance of many subspace clustering (SC) algorithms heavily depends on the availability of held-out datasets for hyperparameters tuning. To address this limitation, we propose a label-independent approach to HPO optimization in existing SC methods. This approach leverages clustering quality metrics such as ACC or NMI, computed in a self-guided manner from pseudo-labels generated by the clustering algorithm over a predefined grid. Additionally, we advocate for visualizing the clustering partitions produced by the proposed method to enable domain experts to asses clustering quality and, if necessary, guide the refinement of the hyperparameters search space. These aspects represent strengths of our HPO method. One limitation of proposed approach lies in its reliance on the initial assessment of the hyperparameter search space. If the initial hyperparameters are set "too closely" together, the method may be misled. While the visualization feature empowers domain experts to refine the hyperparameters search

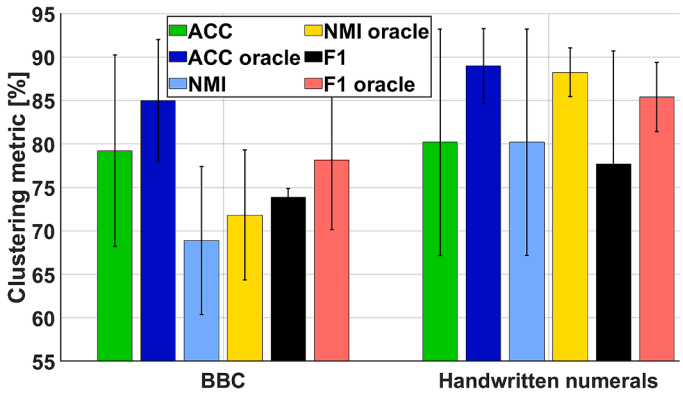


Fig. 10. The clustering performance of the oracle and LFSG versions of the MLME SC algorithm.

space, it requires their involvement, thereby increasing the method's overall computational complexity. Another potential weakness is the method's dependence on the smoothness assumption of clustering quality metrics. Although sensitivity analyses in several studies supports the validity of this assumption, it may not hold for a certain combination of SC algorithms and datasets. In such cases, proposed method may not perform satisfactorily. To enhance robustness against suboptimal initial search space configurations, we aim to evaluate the quality of the pseudo-labels generated by the method. One potential solution involves using non-parametric clustering algorithms, such as *k*-means, to produce the "target" pseudo-labels. While not highly accurate, they can reflect the underlying partition structure of the dataset, enabling our LFSG SC method to identify initial points in the hyperparameter search space that best align with the "target" labels. Another promising direction is to explore metrics related to intra-cluster compactness and inter-cluster separability to select partition corresponding to optimal hyperparameter values. For cases where the data distribution may violate the smoothness assumption, we consider methods for integration of crowdsource labels. This approach treats each set of pseudo-labels generated by an SC algorithm under a specific hyperparameter setting, as an annotation. Methods such as [37] can then estimate the reliability of each "annotator", enabling weighted majority voting to infer a consensus label estimate. Due to space limitation, we did not validate our approach on deep subspace clustering (DSC) methods such as [38, 39]. It is, however, clear that procedure described in section 3.2 can perform tuning of the two hyperparameters of the DSC [38,39]. We anticipate that the LFSG SC method has significant potential to repurpose existing SC algorithms for new domains with unlabeled data, effectively addressing the challenge of label dependency.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author used ChatGPT 3.5 in order to improve the readbility and language of the paper.

CRediT authorship contribution statement

Ivica Kopriva: Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere. We declare that we do not have any known competing financial interests

of personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Croatian Science Foundation under the project number HRZZ-IP-2022-10-6403.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2025.112618.

Data availability

https://github.com/ikopriva/LFSGSC (Data are commonly used datasets. The are availabe at:)

References

[1] C.-N. Jiao, J. Shang, F. Li, et al., Diagnosis-guided deep subspace clustering association study for pathogenetic markers identification of Alzheimer's disease based on comparative atlases, IEEE J. Biomed. Health Inf. 28 (5) (2024) 3029–3041.
[2] A.F. Møller, J.G.S. Madsen, JOINTLY: interpretable joint clustering of single-cell transcriptomes, Nat. Comm. 14 (2023) 8473.
[3] Z. Li, X. He, J. Whitehill, Compositional clustering: applications to multi-label object recognition and speaker identification, Pattern. Recognit. 144 (2023) 109829.
[4] L. Ding, C. Li, D. Jin, S. Ding, Survey of spectral clustering based on graph theory, Pattern. Recognit. 151 (2024) 110366.
[5] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 2765–2781.
[6] M. Brbić, I. Kopriva, *l*o motivated low-rank sparse subspace clustering, IEEE Trans. Cybern. 50 (4) (2020) 1711–1725.
[7] C.-Y. Lu, H. Min, Z.-Q. Zhao, et al., Robust and efficient subspace segmentation via least squares regression. Computer Vision-ECCV 2012, 2012, pp. 347–360.
[8] M. Brbić, I. Kopriva, Multi-view low-rank sparse subspace clustering, Pattern. Recognit. 73 (2018) 247–268.
[9] S. Wang, X. Liu, X. Zhu, et al., Fast parameter-free multi-view subspace clustering with consensus anchor guidance, IEEE Trans. Image Process 31 (2022) 556–568.
[10] Z. Kang, W. Zhu, Z. Zhao, et al., Large-scale multi-view subspace clustering in linear time, in: Proc. of The Thirty Four AAAI Conference on Artificial Intelligence (AAAI-2020), 2020, pp. 4412–4419.
[11] V.M. Patel, R. Vidal, Kernel sparse subspace clustering, in: IEEE International Conference on Image Processing (ICIP), 2014, pp. 2849–2853.
[12] Z. Ma, Z. Kang, G. Luo, et al., Towards clustering-friendly representations: subspace clustering via graph filtering, in: Proceedings of the Media Interpretation & Mobile Multimedia (MM'20), 2020, pp. 3081–3089.
[13] P.A. Traganitis, G.B. Giannakis, Sketched subspace clustering, IEEE Trans. Sig. Proc. 66 (7) (2018) 1663–1675.
[14] J. Xu, Y. Ren, H. Tang, et al., Self-supervised discriminative feature learning for deep multi-view clustering, IEEE Trans. Knowl. Data Eng. 35 (7) (2023) 7470–7482.
[15] J. Gui, T. Chen, Q. Cao, et al., A survey of self-supervised learning from multiple perspectives: algorithms, theory, applications and future trends, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 9052–9071.
[16] G. Zhong, C.-M. Pun, Self-taught multi-view spectral clustering, Pattern. Recognit. 138 (2023) 109349.
[17] J. Lipor, L. Balzano, Clustering quality metrics for subspace clustering, Pattern. Recognit. 104 (2020) 107328.
[18] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, et al., An extensive comparative study of cluster validity indices, Pattern. Recognit. 46 (2013) 243–256.
[19] P.S. Bradley, O.L. Mangasarian, *k*-plane clustering, J. Glob. Optim. 16 (2000) 23–32.
[20] L. Deng, M. Xiao, A new automatic hyperparameter recommendation approach under low-rank tensor completion framework, IEEE Trans. Pattern Anal. Mach. Intell. 45 (4) (2023) 4038–4050.
[21] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (1) (2012) 281–305.
[22] J. Vanschoren, Meta-learning, in: Proc. Automa. Mach. Learn, 2019, pp. 35–61.
[23] S. Gandy, B. Recht, I. Yamada, Tensor completion and low-rank tensor recovery via convex optimization, Inverse Probl. 27 (2) (2011) 025010.
[24] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nat. Mach. Intell. 1 (2019) 206–215.
[25] S. Boyd, N. Parikh, E. Chu, et al., Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn 3 (1) (2011) 1–122.
[26] M. Amini, N. Usunier, C. Goutte, Learning from multiple partially observed views - an application to multilingual text categorization, in: Proc. Adv. Neural Inf. Process. Syst, 2009, pp. 28–36.
[27] W. Hao, S. Pang, B. Yang, J. Xue, Tensor-based multi-view clustering with consistency exploration and diversity regularization, Knowl. Based Syst 252 (2022) 109342.
[28] G. Zhong, C.M. Pun, Improved normalized cut for multi-view clustering, IEEE Trans. Pattern Anal. Mach. Intell. 44 (12) (2022) 10244–10251.
[29] K.-C. Lee, J. Ho, D. Kriegman, Acquiring linear subspaces for face recognition under variable lighting, IEEE Trans. Pattern Anal. Mach. Intell 27 (5) (2005) 684–698.
[30] T. Hastie, P.Y. Simard, Metrics and models for handwritten character recognition, Stat. Sci. 13 (1) (1998) 54–65.
[31] J.J. Hull, A database for handwritten text recognition research, IEEE Trans. Pattern Anal. Mach. Intell. 16 (5) (1994) 550–554.
[32] M. Ribeiro, S. Singh, C. Guestrin, Why should I trust you?: explaining the predictions of any classifier, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, 2016, pp. 97–101.
[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[34] A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 643–660.
[35] F.S. Samaria, A.C. Harter, Parameterisation of a stochastic model for human face identification, in: Proc. 1994 IEEE Workshop on Applic. Comp. Vis. (WACV IEEE), 1994, pp. 138–142.
[36] S.A. Nene, S.K. Nayar, H. Murase, Columbia object image library (coil-100), Tech. Report, CUCS-006-96, Dept. of Computer Science, Columbia Univ. https://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php, 1996.
[37] S. Ibrahim, X. Fu, N. Kargas, K. Huang, Crowdsourcing via pairwise co-occurrences: identifiability and algorithms. Advances in Neural Information Processing Systems (NIPS), 2019, pp. 7847–7857.
[38] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks. Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 24–33.
[39] Y. Akkem, S.K. Biswas, Analysis of an intellectual mechanism of a novel crop recommendation system using improved heuristic algorithm-based attention and cascaded deep learning network, IEEE Trans. Artiif. Intell. 6 (2025) 1100–1113.