Adaptive online k-subspaces with cooperative re-initialization

Connor Lane, Benjamin D. Haeffele, and René Vidal

Mathematical Institute for Data Science, Johns Hopkins University, Baltimore, MD, USA

Abstract

We propose a simple but principled cooperative re-initialization (CoRe) approach to k-subspaces, which also applies to k-means by viewing it as a particular case. CoRe optimizes an ensemble of identical k-subspace models and leverages their aggregate knowledge by greedily exchanging clusters throughout optimization. Further, we introduce an adaptive k-subspaces formulation with split low-rank regularization designed to adapt both the number of subspaces and their dimensions. Moreover, we present a highly scalable online algorithm based on stochastic gradient descent. In experiments on synthetic and real image data, we show that our proposed CoRe method significantly improves upon the standard probabilistic farthest insertion (i.e. k-means++) initialization approach—particularly when k is large. We further demonstrate the improved robustness of our proposed formulation, and the scalability and improved optimization performance of our SGD-based algorithm.

1. Introduction

In this work we study the problem of clustering high-dimensional data drawn from a union of low-dimensional subspaces [32]. The most effective methods for subspace clustering are based on the self-expressive principle: data points belonging to the same subspace can be well represented as linear combinations of one another [11, 12, 24, 22, 33, 36, 41, 42]. These methods exploit this property to first build a graph affinity matrix that is likely to have no false connections between groups, and then apply spectral clustering [35]. However, a significant limitation of these methods is that they do not scale well to very large datasets, since computing the self-expressive matrix is typically at least an $O(N^2)$ operation. On the other hand, k-subspaces, which generalizes k-means, is a highly scalable, linear time alternative [31]. But the corresponding optimization problem is non-convex and suffers from many bad local minima. In addition, the method requires knowledge of the true number of subspaces and their dimensions.

Paper Contributions. In this paper, we introduce a cooperative re-initialization (CoRe) approach to k-subspaces. The main idea is to simultaneously optimize an ensemble of identical k-subspace models (replicas), greedily swap clusters between replicas to improve their quality, and iterate this process till replicas cannot be further improved. By sharing knowledge in this way, we expect the collective performance of the ensemble to gradually improve.

Importantly, the CoRe method can also be applied to k-means, which is in fact a particular case of k-subspaces. In this context, we present CoRe as an alternative to the popular k-means++ initialization scheme [3]. In contrast with k-means++, CoRe is designed to iteratively re-initialize throughout optimization. It is therefore particularly well suited to the large-scale & online settings, where it is impossible or impractical to access the entire dataset up front.

To address the model selection problem, we extend the k-subspaces formulation by adding low-rank regularization to the subspace bases [9, 28]. Crucially, we split the regularization into two terms. In one term, the penalty is scaled relative to cluster size, while in the other the scale is fixed. This enables the regularization to control both the number of subspaces and their dimensions simultaneously.

To further improve the scalability of our method, we implement an online algorithm based on stochastic gradient descent (SGD). Importantly, we exploit the simple structure of our k-subspaces problem to improve upon general SGD. This approach is inspired by the k-GROUSE algorithm, developed for a closely related k-subspaces formulation [4].

Experiments on synthetic data demonstrate the effectiveness of each component of our method. In the k-means setting, we show that CoRe improves upon k-means++ when k is large. We observe a similar pattern with unions of subspaces. Interestingly however, we also observe a significant performance improvement for SGD optimization vs standard Lloyd alternating minimization. When the number of subspaces and their dimensions are unknown, we are nonetheless able to achieve good clustering performance by using split low-rank regularization. Moreover, this performance is robust to the particular choice of regularization parameter value. Last, we demonstrate competitive clustering performance on large-scale real image datasets with up to a 200x speedup compared to previous scalable methods.

2. Problem formulation and related work

Notation. Matrices are denoted with upper case bold letters $\boldsymbol{X}$; vectors with lower case bold letters $\boldsymbol{v}$; scalars with lower case letters α; and arbitrary sets with calligraphic letters $\mathcal{S}$. Sequences are denoted $\{\boldsymbol{x}_i\}_{i=1}^N$ or $\{\boldsymbol{x}_\tau\}_{\tau \in \mathcal{I}}$ for arbitrary index sets $\mathcal{I}$. We will sometimes omit the indexing range or index set when it is clear from the context. $\mathbb{R}^D$ denotes the set of real vectors of dimension D; $\mathbf{0}_D \in \mathbb{R}^D$ the zero vector; $\boldsymbol{I}_D \in \mathbb{R}^{D\times D}$ the identity matrix; $[k]$ the set of integers $1,\dots,k$; Δ^k the standard simplex in k dimensions. Finally, $\|\boldsymbol{X}\|_F$, $\|\boldsymbol{X}\|_*$, and $\|\boldsymbol{X}\|_2$ refer to the Frobenius norm, nuclear norm (also known as trace norm), and spectral norm, respectively.

Problem statement. Let $\mathcal{S}_1,\dots,\mathcal{S}_k$ be a collection of k linear or affine subspaces lying in $\mathbb{R}^D$, each of dimension $d_j < D$. We assume we are given a collection of data points $\mathcal{X} \doteq \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^D$ of size N, where each $\boldsymbol{x}_i \in \mathcal{X}$ is drawn from a distribution $\mathcal{D}_j$ concentrated on $\mathcal{S}_j$. For example, each $\boldsymbol{x}_i$ could be sampled according to a noisy degenerate spherical Gaussian model

$$\begin{gathered} \boldsymbol{x}_i = \boldsymbol{U}_j \boldsymbol{v}_{ij} + \boldsymbol{z}_i,\ \ \boldsymbol{U}_j^\top \boldsymbol{U}_j = \boldsymbol{I}_d, \\ \boldsymbol{v}_{ij} \sim \mathcal{N}(\mathbf{0}_{d_j}, d_j^{-1}\boldsymbol{I}_d),\ \ \boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}_D, \sigma^2 D^{-1}\boldsymbol{I}_D). \end{gathered} \tag{1}$$

The goal of subspace clustering is to identify for each $\boldsymbol{x}_i$, the subspace assignment $\boldsymbol{\alpha}_i \in \Delta^k$ such that $\alpha_{ij} = 1$ if $\boldsymbol{x}_i \sim \mathcal{D}_j$, and zero otherwise.

Self-expressive subspace clustering. Self-expressive methods are currently the most effective approach to the subspace clustering problem [34]. These methods include sparse subspace clustering (SSC) [11, 12], least-squares regression (LSR) [24], low-rank representation (LRR) [22], low-rank subspace clustering [33], elastic-net subspace clustering (EnSC) [41], as well as many others [36]. All of these methods seek to solve a convex optimization problem of the form

$$\underset{\boldsymbol{C}\in\mathbb{R}^{N\times N}}{\text{minimize}} \ \ \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{C}\|_F^2 + \Theta(\boldsymbol{C}), \tag{2}$$

where $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ is the data arranged in a matrix, $\boldsymbol{C}$ is a matrix of self-expressive coefficients, and Θ is a regularization function designed to promote $\boldsymbol{C}$ to be *subspace-preserving*, i.e. $c_{ij} = 0$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to different subspaces. These methods enjoy both strong theoretical guarantees and state-of-the-art performance in practice. However, the quadratic cost of computing the $N \times N$ self-expressive matrix $\boldsymbol{C}$ and applying spectral clustering significantly limits the methods' scalability.

Recently, efficient algorithms with theoretical guarantees have been developed for SSC and EnSC [41, 42]. For example, You and colleagues [41] introduced an efficient active set algorithm to solve the Lasso sub-problem appearing in SSC and EnSC. As a result, their EnSC algorithm achieves dramatically faster runtimes and lower memory cost compared to previous methods. Crucially however, their algorithm still has $O(N^2)$ computational cost, and requires access to the full dataset in memory.

Dictionary based subspace clustering. Another strategy for scalable subspace clustering is to compute $\boldsymbol{C}$ from a dictionary that is much smaller than the dataset. For example, [27, 38] randomly subsample the data, [30] uses a sketched dictionary generated by taking random linear combinations of the data points, while [1] uses a dictionary learned using sparse dictionary learning. Recently, [39] introduced exemplar-based subspace clustering (ESC), which selects exemplar data points using farthest-first search, where the "distance" is measured by the SSC objective.

Crucially, the size of the dictionary used by these methods is often independent of N, which avoids the quadratic cost of the "complete" self-expressive methods. A remaining question is whether the dictionary methods build in enough prior knowledge about the problem to compensate for ignoring some of the data. For example, data that are representable using sparse codes include union of subspaces data as a special case. A sparse dictionary learning approach to subspace clustering then might be too flexible.

k-subspaces. k-subspaces is an alternative subspace clustering method that is at the same time more scalable than self-expressive methods, and with stronger prior knowledge than dictionary based methods [31, 44, 45, 7, 2, 18, 17]. k-subspaces is based on solving the optimization problem:

$$\begin{gathered} \underset{\boldsymbol{\alpha}, \{(\boldsymbol{U}_j, \boldsymbol{b}_j)\}_{j=1}^k}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^k \alpha_{ij} \left(\min_{\boldsymbol{v}} \|\boldsymbol{x}_i - (\boldsymbol{U}_j \boldsymbol{v} + \boldsymbol{b}_j)\|_2^2 \right) \\ \text{s.t.} \ \ \boldsymbol{U}_j^\top \boldsymbol{U}_j = \boldsymbol{I}_{d_j},\ \boldsymbol{\alpha}_i \in \Delta^k, \end{gathered} \tag{3}$$

where $\boldsymbol{U}_j \in \mathbb{R}^{D\times d_j}$ is an orthogonal basis for the jth subspace, $\boldsymbol{b}_j \in \mathbb{R}^D$ is the jth subspace center, and $\boldsymbol{\alpha} \in \mathbb{R}^{N\times k}$ is an assignment matrix whose rows $\boldsymbol{\alpha}_i$ are constrained to lie in the standard simplex Δ^k. Note that although soft assignments are technically permitted, the optimal $\boldsymbol{\alpha}_i$ are guaranteed to be one-hot vectors. This simplex constraint will be assumed throughout and often omitted to reduce notation. Furthermore, $\boldsymbol{v}$ represents a vector of subspace coefficients. Note that including these explicit $\boldsymbol{v}$ coefficients is unnecessary, since they can be computed by orthogonal projection (and indeed this is the more standard formulation). They will be important however in describing the more general formulations to follow.

Notice that k-means clustering can be viewed as a special case of (3) where all subspaces are 0-dimensional. We therefore restrict our discussion to k-subspaces for the majority of this work, although all algorithms apply just as well to k-means. More broadly, in the sequel we consider generalizations of (3) of the form

$$\underset{\boldsymbol{\alpha}, \{(\boldsymbol{U}_j, \boldsymbol{b}_j)\}_{j=1}^k}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{j=1}^{k} \alpha_{ij} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{U}_j, \boldsymbol{b}_j) + \Theta(\boldsymbol{U}_j) \right], \quad (4)$$

where $\mathcal{L}$ is an *assignment loss* and Θ is an *outside regularizer*, as it is not weighted by the assignment variable $\boldsymbol{\alpha}$. The term inside the bracket is the ith term of the objective, and will be denoted F_i. We write the full objective as $F(\boldsymbol{\alpha}, \{(\boldsymbol{U}_j, \boldsymbol{b}_j)\}_{j=1}^k)$, and often $F(\{(\boldsymbol{U}_j, \boldsymbol{b}_j)\}_{j=1}^k)$ to imply an optimal assignment given the $\boldsymbol{U}_j$, $\boldsymbol{b}_j$. Similarly, we will sometimes consider F applied to arbitrary finite basis sets $F(\{(\boldsymbol{U}_\tau, \boldsymbol{b}_\tau)\}_{\tau \in \mathcal{I}})$.

The standard approach to solving (3) in both the k-means and k-subspaces cases is the Lloyd alternating minimization algorithm [23]. After initializing the k subspace bases, $\{(\boldsymbol{U}_j, \boldsymbol{b}_j)\}$, in some way, one alternates between computing the optimal assignment $\boldsymbol{\alpha}$ given the current bases, followed by computing the optimal basis for each of the resulting groups. Crucially however, the k-subspaces objective is non-convex and is known to suffer from many poor local minima. As a result, the Lloyd algorithm is very sensitive to the choice of initialization.

k-subspaces and k-means initialization. The most widely used initialization scheme for k-means and k-subspaces is probabilistic farthest insertion (PFI), popularized by k-means++ [3]. This algorithm initializes each cluster basis sequentially. At each iteration it samples a poorly represented data point $\hat{\boldsymbol{x}}$ to serve as the center for the next cluster. In the k-means setting, one simply sets the current $\boldsymbol{b}_j$ to be $\hat{\boldsymbol{x}}$. In k-subspaces one must initialize $\boldsymbol{U}_j$ as well. A common approach is to perform PCA in the neighborhood of the selected data point [45, 44].

One limitation of PFI is that it is unable to correct poorly initialized clusters. To address this issue one can consider iterative re-initialization schemes, such as the "swap-based" re-initialization strategies, which are particularly relevant to our work [13, 14, 20]. Throughout optimization, these methods seek to identify bad clusters and "swap" them with potentially better alternatives. Strategies for choosing the replacement cluster prototypes include random selection from the data or duplication of an existing cluster. These are heuristic methods however, with no explicit connection to minimizing the objective. Moreover, to the best of our knowledge, no existing swap method leverages the aggregate knowledge of an ensemble to re-initialize.

Another closely related family of methods are the genetic clustering algorithms [5]. These methods do in fact rely on exchanging information among the members of an ensemble to improve performance, much like our proposed method. Often however a heuristic or stochastic criterion is used to guide the "crossover" process. For example, in the classic work of [26], random subsets of clusters are exchanged between selected "chromosomes" (i.e. replicas). By contrast, our proposed method for cluster exchange is guided by decreasing an objective function.

At least one prior work on k-subspaces has also considered an ensemble approach. In [21], Lipor and colleagues introduce ensemble k-subspaces (EKSS), which follows the consensus clustering framework [16]. EKSS runs Lloyd k-subspaces on multiple random initializations, and combines the resulting clusterings through a co-occurrence affinity matrix, which is then clustered by spectral clustering. Two limitations of this approach are (1) many initializations are needed to accumulate enough information about the underlying clusters (often up to 1000, compared to only 8 used in our experiments), and (2) there is still an $O(N^2)$ cost to computing and representing the dense affinity matrix.

3. Description of the proposed method

3.1. Cooperative re-initialization

The first component of our CoRe method is to simultaneously optimize an ensemble of R identical k-subspace replicas, each starting from a different random initialization. Intuitively, we expect that at convergence each replica's clustering will be at least partially correct and that not all replicas will make the same pattern of errors. To exploit this aggregate knowledge across the ensemble, we optimize each replica $r_0 \in [R]$ until its progress slows and then re-initialize it by performing several greedy cluster swaps. More precisely, we attempt to re-initialize a replica r_0 as soon as its objective value has failed to decrease by a fraction δ_{try} for at least T_{try} iterations (Algorithm 2, line 13). At each swap step, we replace one of replica r_0's subspace bases with a better alternative from one of its sibling replicas. Specifically, we first identify the particular subspace basis from the pool of $(R-1)k$ candidates, across all other replicas, whose addition to r_0's clustering would produce the greatest objective decrease. After adding that candidate basis to the clustering, we then identify the basis in r_0 whose removal would result in the smallest objective increase (possibly the same basis that was just added, if no improvement is possible). Finally, we accept the cluster swap if sufficient overall objective decrease is observed.

The CoRe algorithm itself is outlined formally in Algorithm 1. The iterates of the algorithm are sets of replica index, cluster index pairs $\Omega \subset [R] \times [k]$, or "clusterings". Given a complete set of Rk subspace bases $\{(\boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})\}_{r,j=1}^{R,k}$, where r denotes the replica index and j denotes the cluster index within replica r, we define the objective value of the subset of cluster bases indexed by Ω to be $F(\Omega) \triangleq F(\{(\boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})\}_{(r,j)\in\Omega})$. At each iteration, the objective function is evaluated first on $(R-1)k$ candidate clusterings of size $k+1$ to find the best cluster to add (line 5), then on $k+1$ clusterings of size k to determine the best cluster to drop (line 7). Note that clusters from the same replica r_0 need not be considered in line 5, since such additions do not reduce the objective.

Computational complexity. To avoid a dependence on the number of data points N, the objective function in Algorithm 1 is evaluated only on a data subset of size n_{swap}. In the online optimization setting (Section 3.3), we maintain a cache of recently accessed points, while in the batch setting the subset is sampled uniformly at random from the full dataset on each CoRe attempt. By first evaluating and caching the assignment loss $\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})$ and outside regularization $\Theta(\boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})$ for every r, j and $\boldsymbol{x}_i$ in the cache, the two primary steps of the algorithm (lines 5, 7) can be computed using $O(n_{\text{swap}}Rk)$ time and space. We typically choose the maximum number of swap steps, T_{swap}, to be $O(k)$, n_{swap} to be $O(1)$, and R to be $O(\log k)$, resulting in overall time complexity $O(k^2 \log k)$. By contrast, initializing k bases by standard PFI costs $O(Nk^2)$ operations (ignoring dependencies on the ambient and subspace dimensions). Our CoRe method therefore trades the large up-front cost of PFI for a much smaller online cost.

3.2. Split low-rank regularization for adapting k, d

The original k-subspaces formulation requires knowing the exact number of subspaces k and their dimensions $d_1, \dots, d_k$ in advance. Here we introduce a modified formulation, split low-rank k-subspaces (SLR-KSS), designed to adapt these parameters through regularization. As a result, our method can tolerate over-estimates $\hat{k} \geq k$ and $\hat{d} \geq \max_j d_j$. Our formulation is as follows.

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \{\boldsymbol{U}_j, \boldsymbol{b}_j\}}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{j=1}^{\hat{k}} \alpha_{ij} \Big(\min_{\boldsymbol{v}} \|\boldsymbol{x}_i - (\boldsymbol{U}_j \boldsymbol{v} + \boldsymbol{b}_j)\|_2^2 \\ + \|\boldsymbol{v}\|_2^2 + \lambda^{\text{in}} \|\boldsymbol{U}_j\|_F^2 \Big) + \lambda^{\text{out}} \|\boldsymbol{U}_j\|_F^2 \Big], \end{aligned} \tag{5}$$

where now $\boldsymbol{U}_j \in \mathbb{R}^{D \times \hat{d}}$ for all j. In the notation of (4), the term inside the parentheses is the assignment loss $\mathcal{L}$ and $\Theta(\boldsymbol{U}_j) = \lambda^{\text{out}} \|\boldsymbol{U}_j\|_F^2$. Compared to (3) we have replaced the orthogonality constraint on $\boldsymbol{U}_j$ with Frobenius-squared regularization and added ℓ_2-squared regularization on $\boldsymbol{v}$. To motivate this choice, let N_j be the size of cluster j and define a matrix $\boldsymbol{V}_j \in \mathbb{R}^{\hat{d} \times N_j}$ whose columns are the coefficient vectors $\boldsymbol{v}$ corresponding to the points assigned to cluster j. The proposed regularization then merely penalizes the sum of squared Frobenius norms for $\boldsymbol{U}_j$ and $\boldsymbol{V}_j$. This is a simple and standard approach to controlling the rank of the product $\boldsymbol{U}_j \boldsymbol{V}_j$ by penalizing its nuclear (trace) norm through its variational form (see Proposition 3.1) [28, 9]. Our main contribution here is to include the seemingly redundant "split regularization" terms corresponding to λ^{in} and λ^{out}, which we argue help our method adapt to the subspace dimensions and number of subspaces, respectively.

Algorithm 1 Cooperative re-initialization (CoRe)

```
1: Input: replica $r_0 \in [R]$; max swap steps $T_{\text{swap}} > 0$;
   objective decrease accept tolerance $0 < \delta_{\text{accept}} < 1$
2: Output: Updated replica clustering $\Omega_{r_0}^{(s)}$
3: $\Omega_{r_0}^{(0)} \leftarrow \{r_0\} \times [k]$
4: for $s = 1, \dots, T_{\text{swap}}$ do
       ▷ Add cluster giving most objective decrease
5:     $\hat{r}, \hat{\jmath} \leftarrow \arg\min_{(r,j) \in [R] \times [k]} F(\Omega_{r_0}^{(s-1)} \cup \{(r,j)\})$
6:     $\widehat{\Omega}_{r_0}^{(s)} \leftarrow \Omega_{r_0}^{(s-1)} \cup \{(\hat{r}, \hat{\jmath})\}$
       ▷ Drop cluster giving least objective increase
7:     $\bar{r}, \bar{\jmath} \leftarrow \arg\min_{(r,j) \in \widehat{\Omega}_{r_0}^{(s)}} F(\widehat{\Omega}_{r_0}^{(s)} \setminus \{(r,j)\})$
8:     $\bar{\Omega}_{r_0}^{(s)} \leftarrow \widehat{\Omega}_{r_0}^{(s)} \setminus \{(\bar{r}, \bar{\jmath})\}$
       ▷ Accept swap if sufficient overall objective decrease
9:     if $F(\bar{\Omega}_{r_0}^{(s)}) \leq (1 - \delta_{\text{accept}}) F(\Omega_{r_0}^{(s-1)})$ then
10:        $\Omega_{r_0}^{(s)} \leftarrow \bar{\Omega}_{r_0}^{(s)}$
11:    else: $\Omega_{r_0}^{(s)} \leftarrow \Omega_{r_0}^{(s-1)}$; break
```

To better understand the effect of this regularization, consider the optimization problem with respect to $\boldsymbol{U}_j$, $\boldsymbol{b}_j$, and the newly defined $\boldsymbol{V}_j$ for a fixed group j. Assume that the subset of the data assigned to group j, $\boldsymbol{X}_j \in \mathbb{R}^{D \times N_j}$, is centered so that we may ignore $\boldsymbol{b}_j$. After re-arranging (5), this restricted optimization problem can be written as

$$\begin{aligned} \underset{\boldsymbol{U}_j, \boldsymbol{V}_j}{\text{minimize}} \ \|\boldsymbol{X}_j - \boldsymbol{U}_j \boldsymbol{V}_j\|_F^2 + (\lambda^{\text{in}} N_j + \lambda^{\text{out}} N) \|\boldsymbol{U}_j\|_F^2 \\ + \|\boldsymbol{V}_j\|_F^2. \end{aligned} \tag{6}$$

To further characterize the problem, we give the following proposition based on standard facts (see e.g. [10]).

Proposition 3.1. *The restricted problem (6) is equivalent to the nuclear norm proximal optimization problem*

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \frac{1}{2} \|\boldsymbol{X}_j - \boldsymbol{W}\|_F^2 + \tau_j \|\boldsymbol{W}\|_*, \tag{7}$$

with $\tau_j = (\lambda^{in} N_j + \lambda^{out} N)^{1/2}$. *Also, a solution to* (7) *is given by* $\boldsymbol{W}^* = \boldsymbol{P}(\Sigma - \tau_j \boldsymbol{I})_+ \boldsymbol{Q}^\top$, *where* $\boldsymbol{X}_j = \boldsymbol{P} \Sigma \boldsymbol{Q}^\top$ *is the SVD and* $(\boldsymbol{X})_+ = \max(\boldsymbol{X}, 0)$ *is applied entrywise. Consequently, a solution to (6) is* $\boldsymbol{U}^* = P(\Sigma - \tau_j \boldsymbol{I})_+^{1/2}$, $\boldsymbol{V}^* = (\Sigma - \tau_j \boldsymbol{I})_+^{1/2} \boldsymbol{Q}^\top$, *provided* $\hat{d} \geq \operatorname{rank}(\boldsymbol{W}^*)$.

Thus, the proposed Frobenius-squared regularization controls the ranks of the subspaces through singular value soft-thresholding. However, note that λ^{in} and λ^{out} contribute differently to the threshold τ_j. Because the λ^{in} term is placed *inside* the assignment loss in (5), its effect depends on the size of the jth group, N_j. On the other hand, the impact of the λ^{out} term is independent of N_j.

To first understand the impact of λ^{in}, suppose $\lambda^{\text{out}} = 0$. For typical datasets, we expect the singular values of $\boldsymbol{X}_j$ to scale like $O(\sqrt{N_j})$ and for there to exist a gap after the

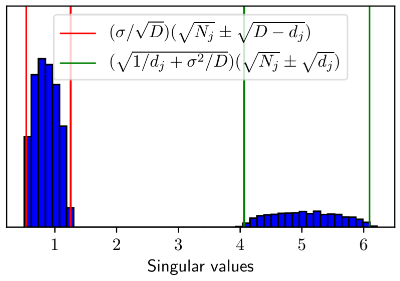

Figure 1. Singular value histogram for synthetic Gaussian data generated according to (1) with $d = 20$, $D = 100$, $N_j = 500$, and $\sigma = 0.4$. The red and green lines indicate the "noise" and "data" singular value intervals respectively. Best viewed in color.

d_jth largest singular value. For example, in the degenerate spherical Gaussian setting (1), we know with high probability that the top d_j "data" singular values will lie in the interval $(\sqrt{1/d_j + \sigma^2/D})(\sqrt{N_j} \pm \sqrt{d_j})$, whereas the remaining "noise" singular values will lie in $(\sigma/\sqrt{D})(\sqrt{N_j} \pm \sqrt{D - d_j})$ [15] (Figure 1). There should therefore exist an interval $[\lambda_0^{\text{in}}, \lambda_1^{\text{in}}]$, depending primarily on the noise level of the data, such that if $\lambda_0^{\text{in}} \le \lambda^{\text{in}} \le \lambda_1^{\text{in}}$, the solution to (6) will recover d_j, regardless of the cluster size N_j.

A side-effect of this invariance to N_j, however, is that splitting a cluster can only reduce the objective in (5). Note that when $\lambda^{\text{out}} = 0$, appending a duplicate basis to the current model will not increase the objective. Thus, when k is over-estimated, some $\lambda^{\text{out}} > 0$ is necessary to avoid excessive splitting. Intuitively, the correct choice for λ^{out} should depend on the principal angles between the subspaces and the noise level. The formulation will prefer to merge two closely aligned subspaces if λ^{out} is too large. On the other hand, if the data are noisy, setting λ^{out} too small will result in unnecessary splits. As with λ^{in}, we conjecture that for a given dataset there will exist an interval of λ^{out} values for which one can recover the true number of subspaces.

To simplify the selection of particular λ^{in}, λ^{out} for a given problem, we re-parameterize in terms of two more interpretable quantities, a noise estimate $\hat{\sigma} > 0$ and a "minimum cluster fraction" $\rho \in [0, 1]$, as well as the estimated number of subspaces $\hat{k}$ and their dimension $\hat{d}$, as follows:

$$\begin{aligned}\lambda^{\text{in}} &= (\hat{\sigma}^2/D)\Big(1 + \sqrt{(D-\hat{d})\hat{k}/N}\Big)^2 \\ \lambda^{\text{out}} &= (1/\hat{d} + \hat{\sigma}^2/D)(\rho/\hat{k}).\end{aligned} \tag{8}$$

Applying Proposition 3.1, these choices yield a singular value threshold $\tau_j = (N_j\lambda^{\text{in}} + N\lambda^{\text{out}})^{1/2}$ satisfying

$$\begin{aligned}\tau_j \ge \max\Big\{&(\hat{\sigma}/\sqrt{D})\Big(\sqrt{N_j} + \sqrt{(D-\hat{d})(N_j\hat{k}/N)}\Big), \\ &\Big(\sqrt{1/\hat{d} + \hat{\sigma}^2/D}\Big)\Big(\sqrt{\rho N/\hat{k}}\Big)\Big\}.\end{aligned} \tag{9}$$

Following the previous discussion, the first term represents the estimated right bulk edge of the "noise" singular values, whereas the second term corresponds to the median "data" singular value (again assuming the Gaussian data setting (1)). As a result, when $\hat{\sigma}$ is accurate, we expect to threshold all noise singular values and recover the true dimensions d_j. On the other hand, if a cluster's size approaches $\rho N/k$, we will begin to threshold the data singular values. This should significantly reduce the cluster's ability to represent its assigned data and, we conjecture, cause it to further shrink in size until no assigned points remain and the basis is set to zero.

3.3. An online algorithm based on SGD

A central goal for the current work was to design a highly scalable subspace clustering method. k-subspaces is inherently more scalable than self-expressive methods, since its computational and memory costs are only linear in N compared to quadratic. Here we consider an online algorithm (Algorithm 2) based on stochastic gradient descent (SGD), whose per-iteration computational and memory cost is independent of N.

Algorithm 2 SGD for split low-rank k-subspaces with cooperative re-initialization (CoRe SLR-KSS SGD)

```
1: Input: dataset $\mathcal{X}$; formulation parameters $(\hat{k}, \hat{d}, \lambda^{\text{in}}, \lambda^{\text{out}})$; number of replicas $R$; patience parameters $(T_{\text{try}}, \delta_{\text{try}})$; CoRe parameters $(T_{\text{swap}}, n_{\text{swap}}, \delta_{\text{accept}})$; SGD parameters $(T, n_{\text{bs}}, \eta, \beta, T_{\text{bs}})$
2: Output: Final subspace models for each replica $\{(\boldsymbol{U}_{r,j}^{(T)}, \boldsymbol{b}_{r,j}^{(T)})\}$
3: Initialize subspace models $\{(\boldsymbol{U}_{r,j}^{(1)}, \boldsymbol{b}_{r,j}^{(1)})\}$ with small Gaussian entries.
4: for $t = 1, \dots, T$ do
5:   Sample a data mini-batch $\mathcal{I} \subset [N]$ of size $n_{\text{bs}}$
6:   for $r = 1, \dots, R$ do
7:     Find optimal $\boldsymbol{v}$ by least-squares (for all $j, i \in \mathcal{I}$)
8:     Find optimal assignments $\boldsymbol{\alpha}_{r,i}$
9:     Evaluate $F_{\mathcal{I}}(\{(\boldsymbol{U}_{r,j}^{(t)}, \boldsymbol{b}_{r,j}^{(t)})\})$; update EMA $\bar{F}_r^{(t)}$
10:    Update running Hessian $\boldsymbol{H}_{\boldsymbol{U}_{r,j}}^{(t)}$ based on $F_{\mathcal{I}}$
11:    $\tilde{L}_r^{(t)} = \max_j \| \operatorname{diag}(\boldsymbol{H}_{\boldsymbol{U}_{r,j}}^{(t)})\|_\infty$
12:    SGD$(\eta/L_r^{(t)}, \beta)$ step on $\{(\boldsymbol{U}_{r,j}^{(t)}, \boldsymbol{b}_{r,j}^{(t)})\}$
13:  for $r = 1, \dots, R$ do
14:    if $\bar{F}_r^{(t)}$ has not decreased by $\delta_{\text{try}}$ in $T_{\text{try}}$ iters then
15:      $\Omega_r^{(t)} \leftarrow$ CoRe$(r, T_{\text{swap}}, \delta_{\text{accept}})$
16:      Make swaps in $\{(\boldsymbol{U}_{r,j}^{(t)}, \boldsymbol{b}_{r,j}^{(t)})\}$ according to $\Omega_r^{(t)}$
17:  if $t \bmod T_{\text{bs}} = 0$ then: $n_{\text{bs}} \leftarrow 2n_{\text{bs}}$
```

At each iteration we first sample a data mini-batch $\mathcal{I} \subseteq [N]$ of size n_{bs}. For each replica, we proceed as follows. For each cluster and data point in the batch, we compute the optimal coefficients $\boldsymbol{v}$ exactly by solving a least-squares problem. This is the most computationally expensive step of the algorithm, requiring $O(R\hat{k}\hat{d}^2(D+n_{\text{bs}}))$ operations in total. Next, we assign each data point to the cluster achieving the smallest assignment loss. We then take a stochastic gradient step on the $\{(\boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})\}$ with momentum β. Last, we attempt cooperative re-initialization (Algorithm 1) on any replicas whose progress in objective has slowed. Since we do not have access to exact objective values, we use exponential moving averages (EMAs) $\bar{F}_r^{(t)}$ instead.

To select the stochastic gradient step size, we exploit the relatively simple structure of the k-subspaces objective. The optimization problem with respect to the subspace parameters $(\boldsymbol{U}_{r,j}, \boldsymbol{b}_{r,j})$ is a convex quadratic, with a mini-batch Hessian of the form $\sum_{i\in\mathcal{I}} \boldsymbol{v}_i\boldsymbol{v}_i^\top \in \hat{d}\times\hat{d}$ (up to constants). This Hessian can be computed inexpensively, allowing us to easily estimate step sizes based on the local Lipschitz constant of the gradient. Specifically, we first accumulate the full Hessian $\boldsymbol{H}_{\boldsymbol{U}_{r,j}}$ over the current epoch and one previous, for every replica and cluster, following [25]. We then approximate the local Lipschitz constant as $L_r = \|\boldsymbol{H}_{\boldsymbol{U}_{r,j}}\|_2$ by the maximum diagonal entry $\tilde{L}_r = \max_j \|\operatorname{diag}(\boldsymbol{H}_{\boldsymbol{U}_{r,j}})\|_\infty \leq L_r$. This approximation is cheaper to compute compared to evaluating the spectral norm, and is expected to be accurate since $\boldsymbol{U}_j$ is expected to have nearly orthogonal columns (Proposition 3.1).

In SGD, it is common to reduce the step size during optimization to gradually temper the effect of gradient noise. Alternatively, one can also *increase* the batch size to achieve similar benefits [29]. This latter strategy is well-suited to our case since we know a reasonable step size for the problem a priori. Furthermore, increasing batch size rather than reducing step size significantly improves data throughput when using GPUs, strengthening the scalability of our method. We chose a very simple batch size schedule: doubling n_{bs} every T_{bs} steps until reaching a fixed upper bound depending on memory capacity.

4. Experiments

Across all experiments, we compared two optimization algorithms: Lloyd alternating minimization and SGD (Algorithm 2), as well as two (re-)initialization strategies: PFI and CoRe. The intended regime for our method is large-scale data, where accessing the entire dataset for initialization is impossible or impractical. Consistent with this, both PFI and CoRe were restricted to using data subsets of only 1000 samples. In addition, only 8 initializations/replicas were used in each experiment.

Unless otherwise stated, we optimized for a fixed 50 epochs using step size $\eta = 0.1$, momentum $\beta = 0.9$, initial batch size $n_{\text{bs}} = 50$, and batch size increase period $T_{\text{bs}} = 10$ epochs. We also used a fixed re-initialization tolerance $\delta_{\text{try}} = 0.01$ with patience $T_{\text{try}} = 200$ steps for SGD and $T_{\text{try}} = 2$ steps for Lloyd. We set the number of swap iterations $T_{\text{swap}} = k/2$, and swap accept tolerance $\delta_{\text{accept}} = 0.001$. The values reported in the experiments were taken "at convergence". That is, they correspond to the first epoch achieving an objective value within a factor of 10^{-3} of the lowest observed.

Our primary performance metric is clustering error, defined to be the fraction of incorrectly classified data points, up to permutation of the cluster labels. We used the Hungarian algorithm to compute the optimal matching between cluster labels and true labels. When $\hat{k} \neq k$, the Hungarian algorithm selects optimal label subsets of size $\min\{\hat{k}, k\}$ for both cluster and true labels. The matching is then computed just over these subsets. If a data point's assigned cluster label or true label does not belong to the respective matched subset, it is considered incorrect. We report clustering error in $1/k$ "cluster size" units. A clustering error of 1 therefore indicates approximately 1 cluster's worth of mistakes. For example, a clustering error of 3 for $k = 10$ indicates 30% error.

4.1. Synthetic k-means

We first evaluated our CoRe SLR-KSS SGD algorithm in a synthetic k-means setting with known k and no regularization. Data were generated according to the model

$$\begin{gathered} \boldsymbol{x}_i = \boldsymbol{\mu}_j + \boldsymbol{z} \\ \boldsymbol{\mu}_j \sim \mathcal{N}(\boldsymbol{0}_D, (\psi^2/2)D^{-1}\boldsymbol{I}_D), \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}_D, D^{-1}\boldsymbol{I}_D), \end{gathered} \tag{10}$$

so that in high-dimension, the separation between clusters is $\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_{j'}\|_2 \approx \psi$ and $\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2 \approx 1$ if $\boldsymbol{x}_i$ is drawn from group j. When $\psi < 2$, there can be significant overlaps in distribution between clusters. However, if $k < D$, then in the relevant subspace spanning the k cluster centers, there will be no overlap with high probability for separations as small as $\psi = 2\sqrt{k/D}$ [6].

We performed one experiment varying the number of clusters $k = 10, 20, \ldots, 100$ with fixed $\psi = 2$, and a second experiment fixing $k = 10$ and varying ψ. We fixed the ambient dimension $D = 100$ and number of data points $N = 10,000$. We varied the separation ψ across 10 linearly spaced values within $[\sqrt{k/D}, 1.5]$. Importantly, at $\psi = \sqrt{k/D}$ we expect poor performance due to near total cluster overlap. We let $\hat{k} = k$ and did not include regularization. Finally, we repeated both experiments for 20 trials with different random datasets and initializations.

Figure 2 (left and middle left) reports the mean clustering errors across the two experiments. First, for increasing k we observe steadily increasing errors for both PFI-initialized optimization algorithms. By contrast, both algorithms using CoRe achieve perfect clustering for all values of k. This clearly demonstrates CoRe's ability to iteratively

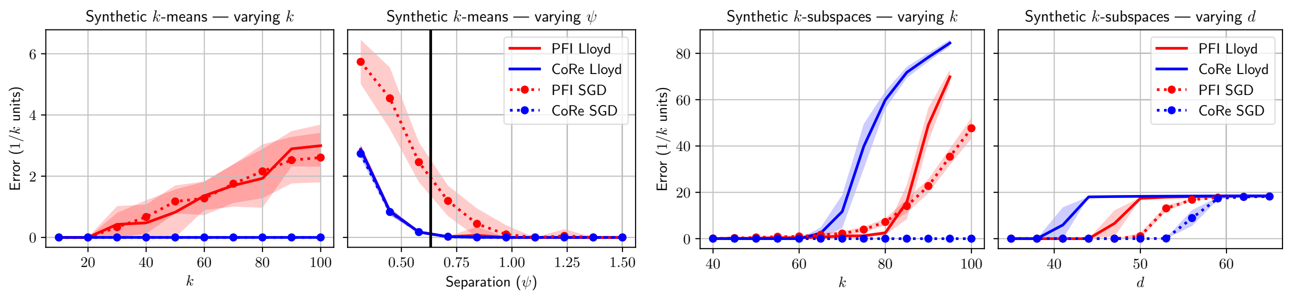

Figure 2. Clustering error on synthetic k-means and k-subspaces data. **(Left)** k-means clustering errors for increasing number of clusters k and well-separated means (separation $\psi = 2.0$). **(Middle left)** k-means clustering errors for increasing cluster separation with fixed $k = 10$. The black vertical line indicates the overlap threshold $\psi = 2\sqrt{k/D}$. **(Middle right)** k-subspaces clustering errors for increasing k, fixed $d = 20$, and fixed noise level $\sigma = 0.4$. **(Right)** k-subspaces clustering errors for fixed $k = 20$, increasing d, and fixed $\sigma = 0.4$. Shaded region represents stdev. Best viewed in color.

correct poorly initialized clusters (which occur more often in PFI with increasing k).

In the second experiment, we observe near perfect clustering performance for all methods except PFI SGD once the overlap threshold $\psi = 2\sqrt{k/D}$ is reached. PFI SGD on the other hand requires greater separation to achieve comparable performance. Intuitively, the gradient noise in SGD should make resolving closely packed clusters more difficult. We implement an increasing batch size schedule partly to overcome this challenge (Section 3.3). Interestingly, this result suggests that increasing batch size alone is not sufficient when PFI is used, and that *iterative* re-initialization with CoRe is beneficial.

4.2. Synthetic k-subspaces with known k, d

We next evaluated our proposed method in a synthetic union of subspaces setting where we assume k and d are known. We generated data according to the degenerate spherical Gaussian model (1) with fixed $D = 100$, $\sigma = 0.4$, and N =10,000. The subspaces were sampled uniformly from the Grassmannian of dimension d by sampling $\boldsymbol{U}_j$ with standard Gaussian entries and orthogonalizing.

As with k-means, we considered two experimental settings for varying clustering difficulty. In the first setting, we varied $k = 40, 45, \ldots, 100$ with a fixed (relatively) small subspace dimension $d = 20$. Importantly, note that the subspaces are non-independent even for the smallest choice of k. In the second setting, we fixed $k = 20$ and varied $d = 35, 38, \ldots, 65$. Since $D = 100$, there will be guaranteed intersections between every pair of clusters as soon as $d > 50$, making clustering significantly more difficult. We let $\hat{k} = k$, $\hat{d} = d$ and did not include regularization. Both experiments were repeated for 30 random trials.

Figure 2 (middle right and right) reports mean clustering errors across the two experiments. The first pattern we observe in both settings is that SGD optimization alone significantly improves clustering performance, across initialization schemes. This is in contrast to the k-means case where the choice of optimization only impacted PFI initialization, and only when varying cluster separation. Second, we note that CoRe performs much worse than PFI for Lloyd optimization, but much better for SGD.

One possible explanation for these observations is that the noise present in SGD optimization helps the replicas escape or avoid poor local minima that Lloyd optimization would otherwise get stuck in. Indeed, it has been argued that SGD is often able to avoid sharp local minima in favor of wider, more robust regions of the optimization landscape [19]. When clusters are first initialized randomly in the CoRe only case and Lloyd optimization is used, the replicas may quickly become stuck in very poor, sharp local minima. Since CoRe depends on the initial clusterings being at least partially correct, it is unable to correct these gross errors. SGD on the other hand might provide enough of a "jump start" to allow CoRe to further refine the clusterings.

Focusing on SGD optimization, we observe results that are largely consistent with the k-means setting. For increasing k, PFI initialization produces significant errors (up to $k/2$ clusters' worth), whereas CoRe remains perfect up to the maximum $k = 100$. This further solidifies CoRe's advantage in the large k regime.

We observe a similar pattern in the varying dimension experiment. All methods undergo a severe phase transition near $d = 50$. The threshold occurs earliest, however, for CoRe Lloyd, and latest for CoRe SGD. Intuitively, PFI initialization is more likely to fail when the subspaces are high-dimensional and overlapping. This is because the local neighborhoods of the candidate insertion centers (used to initialize new bases) are likely to be more sparsely populated and corrupted with outlier points from other groups. CoRe on the other hand should be more robust in this setting, since the candidate re-initialization bases (drawn from other replicas) are fit over the course of optimization to an entire cluster instead of just a neighborhood.

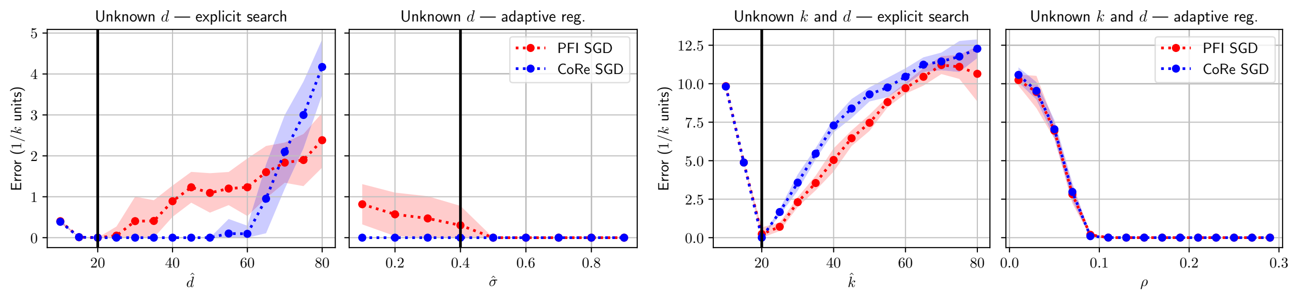

Figure 3. SGD k-subspaces clustering errors on synthetic data when k and d are unknown. **(Left)** Known $\hat{k} = k = 20$ but unknown $d = 20$, evaluated over a range of $\hat{d}$ with no regularization ($\hat{\sigma} = \rho = 0$). True d indicated with a black line. **(Middle left)** Same setting as left but now with fixed $\hat{d} = 60$ and varied "inside" regularization parameter $\hat{\sigma}$ (the approximate noise level). True noise level $\sigma = 0.4$ indicated with a black line. **(Middle right)** Unknown $k = 20$ and unknown $d = 20$ with fixed $\hat{d} = 60$, $\hat{\sigma} = \sigma = 0.4$. Error evaluated over a range of $\hat{k}$ with no "outside" regularization ($\rho = 0$). True k indicated with a black line. **(Right)** Same setting as middle right, but with fixed $\hat{k} = 60$ and varied "outside" regularization parameter ρ (the "minimum cluster fraction"). Shaded region represents stdev. Best viewed in color.

4.3. Adapting to unknown k and d

We next examined how well our proposed split low-rank regularization is able to adapt to unknown k and d on synthetic union of subspaces data. We again generated data according to the Gaussian model (1) with fixed $D = 100$, $\sigma = 0.4$, and N =10,000. In this experiment, we also concentrated on a relatively easy setting from Section 4.2: $k = d = 20$, and restricted to only SGD optimization.

We again considered two experimental settings: first assuming a known k and unknown d, and second assuming both an unknown k and d. In each setting we also evaluated two approaches for adapting to the unknown data parameters: first, searching over the approximate parameters $\hat{k}$, $\hat{d}$ directly, without regularization (explicit search), and second, using fixed over-estimates for $\hat{k}$ and $\hat{d}$ while searching over the regularization parameters $\hat{\sigma}$ and ρ (adaptive regularization). The key question is whether clustering performance is more robust in one approach vs the other.

Note that in (1), the standard deviation of the noiseless data is 1. Thus, a reasonable upper limit is $\hat{\sigma} = 1$, corresponding to noise equal in magnitude to the underlying signal. Moreover, there is a clear "ground truth" choice: $\hat{\sigma} = \sigma$. It is less clear how to choose the minimum cluster fraction ρ, although we do know the acceptable range of values: $\rho \in [0, 1)$.

In the first experimental setting, we consider $\hat{d} = 10, 15, \dots, 80$ and no regularization ($\hat{\sigma} = 0$) for explicit search, and $\hat{d} = 60 = 3d$ with $\hat{\sigma} = 0.1, 0.2, \dots, 1.5$ for adaptive regularization. For both strategies we set $\rho = 0$. In the second experiment, we set an over-estimate $\hat{d} = 60 = 3d$ and fix $\hat{\sigma} = \sigma = 0.4$ for both search strategies. Then for explicit search we look at $\hat{k} = 10, 15, \dots, 80$ and $\rho = 0$, while for adaptive regularization we set $\hat{k} = 60 = 3k$ and vary $\rho = 0.01, 0.03, \dots, 0.29$.

Figure 3 reports mean clustering errors for these experiments. In the first experiment, we find that PFI initialized SGD (without regularization) is moderately sensitive to deviations from the true $d = 20$. For example, for $\hat{d} = 60$, the method suffers up to 2 clusters' worth of errors. Interestingly, CoRe appears significantly less sensitive, achieving near zero error for the same $\hat{d}$. Although for $\hat{d} > 60$ its error increases sharply, ultimately surpassing PFI at $\hat{d} = 60$. It is not entirely clear why CoRe should favor these more accurate solutions. Without regularization, a correct clustering will not produce a significantly smaller objective than, for example, a solution that merges several subspaces into one group. One possibility is that CoRe introduces unaccounted *implicit* regularization that biases the clusterings in this way.

By contrast, for the adaptive regularization strategy we observe stable, near perfect performance across the range of reasonable values $\hat{\sigma} \in (0, 1)$. The only exception is the somewhat worse performance of PFI for smaller $\hat{\sigma}$. This robust performance is strong evidence in favor of the proposed adaptive regularization approach. One caveat however is that clustering performance severely degrades for $\hat{\sigma} \geq 1$, particularly for CoRe initialization (not shown in Figure 3).

This pattern is even clearer in the second experiment where both k and d are unknown. For the explicit search strategy, clustering performance quickly deteriorates as $\hat{k}$ deviates from the true $k = 20$. Whereas for fixed $\hat{k} = 60$, clustering performance is perfect for both methods as long as ρ is above a threshold ≈ 0.1. Importantly, both strategies employ the same level of "inside" regularization, with "outside" regularization included only for the adaptive regularization strategy. This suggests that the two seemingly redundant regularization terms are in fact crucial for simultaneously adapting to k and d.

Table 1. Clustering accuracy

	SLR-KSS SGD		EKSS	EnSC		ESC	
	PFI	CoRe	·	·	k-NN	100	320
EYaleB	62.3	77.0	85.7	67.9	94.5	—	—
COIL100	71.9	82.3	71.4	86.2	60.3	—	—
MNIST	98.5	98.4	97.6	94.0	98.3	—	—
EMNIST	62.0	66.1	—	—	—	66.3	73.4

Table 2. Runtime (seconds)

	SLR-KSS SGD		EnSC		ESC	
	PFI	CoRe	·	k-NN	100	320
EYaleB	307	583	47	60	—	—
COIL100	806	661	34	73	—	—
MNIST	476	556	2234	2264	—	—
EMNIST	113	52	—	—	4015	10837

4.4. Clustering real image data

In a final experiment, we evaluated the clustering performance and runtime of our proposed CoRe SLR-KSS SGD method on four real image datasets: Extended Yale-B face images under varying illumination (EYaleB; $k = 38$, $N = 2,414$), COIL100 objects under rotation ($k = 100$, $N = 7,200$), MNIST digits ($k = 10$, $N = 70,000$), and Extended MNIST lower case letters ($k = 26$, $N = 190,998$). The datasets were pre-processed following [37]. EYaleB images were downsampled to 48×42 and then projected to $D = 500$ by PCA. For all other datasets, we first extracted scattering network features [8], followed by PCA to reduce also to $D = 500$. In addition, on some evaluation trials, we performed a form of pseudo-whitening where 1-2 of the top principal components were removed from the dataset. This procedure has been used previously with k-subspaces, and is argued to help create separation between subspaces with small principal angles [45].

We evaluated SLR-KSS SGD with both initialization schemes over a range of 30 hyper-parameter settings sampled uniformly from a defined grid, and report the best setting based on cluster accuracy. The parameters we considered and their ranges were: the number of principal components removed in whitening $\in \{0, 1, 2\}$, the subspace dimension $\hat{d} \in \{5, 10, 20, 40\}$, the noise estimate $\hat{\sigma} \in \{0.0, 0.1, \dots, 0.6\}$, the step size $\eta \in \{0.01, 0.02, 0.04, \dots, 0.64\}$, and initial batch size $n \in \{20, 40, 80, 160, 320\}$. We optimized for a fixed 100 "epochs" of size $70,000$ samples irrespective of dataset size, and report values corresponding to the first epoch achieving a clustering error at most .1% worse than the best observed. Finally, each run of our evaluation was executed using an Nvidia RTX 2080 GPU with 11GB of memory.

For comparison, we include accuracies and runtimes drawn from the respective papers for three scalable subspace clustering methods, one for each class of methods reviewed in Section 2: ensemble k-subspaces (EKSS) [21], elastic-net subspace clustering (EnSC) [41], and exemplar subspace clustering (ESC) [40]. We include two variants for EnSC: one using the standard affinity construction and another where the self-expression matrix is used to compute a k-nearest neighbor graph. For ESC, we include results for 100 and 300 exemplar data points. The pre-processing pipelines used here are largely identical to those for EnSC and ESC, whereas [21] analyzes a 10,000 sample subset of MNIST and uses slightly different pre-processing steps.

The results of our evaluation are reported in Tables 1 and 2. First, we observe that for the EYaleB, COIL100, and EMNIST datasets, CoRe initialization significantly outperforms PFI (24%, 14%, and 7% relative improvement respectively). On MNIST, there is little room for CoRe to improve, since both initializations meet the state-of-the-art clustering performance. On EMNIST, the CoRe method approaches the previously reported accuracies for ESC-100, but not those of the more accurate (and computationally expensive) ESC-300. On EYaleB and COIL100, our method is able to out-perform some comparisons but not others (although the winning comparison method is different in the two cases). Importantly, these are also small scale datasets which do not represent our intended regime.

In terms of runtime, our proposed methods provide significant speed-ups on the larger datasets compared to the other scalable methods, albeit using more powerful (but not uncommon) compute hardware (4x-10x on MNIST, 10x-77x on EMNIST vs ESC-100, and 30x-210x vs ESC-300). Interestingly, the runtimes on the two smaller datasets are somewhat *slower*. This result suggests that the time needed to reach convergence on these datasets is largely independent of dataset size, as is common for online algorithms.

5. Conclusions

We proposed the cooperative re-initialization (CoRe) method for k-subspaces and k-means as a way to iteratively escape bad local minima throughout optimization. In addition, we introduced an extension of the standard k-subspaces formulation that includes a "split low-rank" regularizer. We also implemented a highly scalable online algorithm based on SGD, but which also takes advantage of the special simple structure of the optimization problem. We validated all three components of our method in synthetic experiments, and demonstrated competitive clustering accuracy and state-of-the-art scalability on large-scale image data. Possible future directions for this work include developing extensions to handling various forms of corrupted data [4], as well as extensions to nonlinear manifolds through integration with deep auto-encoders [43].

Acknowledgements. This work was supported by NSF grants 1618485 and 1704458.

References

[1] A. Adler, M. Elad, and Y. Hel-Or. Linear-time subspace clustering via bipartite graph modeling. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2234 – 2246, 2015.

[2] Pankaj K Agarwal and Nabil H Mustafa. K-means projective clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 155–165. ACM, 2004.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] Laura Balzano, Arthur Szlam, Benjamin Recht, and Robert Nowak. K-subspaces with missing data. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 612–615. IEEE, 2012.

[5] Sanghamitra Bandyopadhyay. Genetic algorithms for clustering and fuzzy clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(6):524–531, 2011.

[6] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. 2016.

[7] Paul S Bradley and Olvi L Mangasarian. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[8] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[9] Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[10] J-F Cai, E. J. Candés, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal of Optimization*, 20(4):1956–1982, 2008.

[11] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.

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

[13] Pasi Fränti. Efficiency of random swap clustering. *Journal of Big Data*, 5(1):13, 2018.

[14] Pasi Fränti and Olli Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–775, 2006.

[15] Matan Gavish and David L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.

[16] Joydeep Ghosh and Ayan Acharya. Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315, 2011.

[17] Andrew Gitlin, Biaoshuai Tao, Laura Balzano, and John Lipor. Improving k-subspaces via coherence pursuit. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1575–1588, 2018.

[18] Jun He, Yue Zhang, Jiye Wang, Nan Zeng, and Hanyong Hao. Robust k-subspaces recovery with combinatorial initialization. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3573–3582. IEEE, 2016.

[19] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[20] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[21] John Lipor, David Hong, Dejiao Zhang, and Laura Balzano. Subspace clustering using ensembles of k-subspaces. *arXiv preprint arXiv:1709.04744*, 2017.

[22] G. Liu, Z. Lin, S. Yan, J. Sun, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[23] S. Lloyd. Least squares quantization in PCM. *Technical Report, Bell Laboratories, Published in 1982 in IEEE Trans. Inf. Theory 28: 128-137*, 1957.

[24] C-Y. Lu, H. Min, Z-Q. Zhao, L. Zhu, D-S. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, pages 347–360, 2012.

[25] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[26] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465, 2000.

[27] Xi Peng, Lei Zhang, and Zhang Yi. Scalable sparse subspace clustering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2013.

[28] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[29] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

[30] Panagiotis A. Traganitis and Georgios B. Giannakis. Sketched subspace clustering. *IEEE Transactions on Signal Processing*, 2017.

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

[32] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(3):52–68, March 2011.

[33] René Vidal and Paolo Favaro. Low rank subspace clustering (LRSC). *Pattern Recognition Letters*, 43:47–61, 2014.

[34] René Vidal, Yi Ma, and Shankar Sastry. *Generalized Principal Component Analysis*. Springer Verlag, 2016.

[35] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[36] Yu-Xiang Wang, Huan Xu, and Chenlei Leng. Provable subspace clustering: When LRR meets SSC. In *Neural Information Processing Systems*, 2013.

[37] Chong You. *Sparse Methods for Learning Multiple Subspaces from Large-scale, Corrupted and Imbalanced Data*. PhD thesis, The Johns Hopkins University, 2018.

[38] C. You, C. Donnat, D. Robinson, and R. Vidal. A divide-and-conquer framework for large-scale subspace clustering. In *Asilomar Conference on Signals, Systems and Computers*, 2016.

[39] Chong You, Chi Li, Daniel P. Robinson, and René Vidal. A scalable exemplar-based subspace clustering algorithm for class-imbalanced data. In *European Conference on Computer Vision*, 2018.

[40] Chong You, Chi Li, Daniel P Robinson, and René Vidal. Scalable exemplar-based subspace clustering on class-imbalanced data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–83, 2018.

[41] Chong You, Chun-Guang Li, Daniel P. Robinson, and René Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3928–3937, 2016.

[42] Chong You, Daniel P. Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3918–3927, 2016.

[43] Tong Zhang, Pan Ji, Mehrtash Harandi, Richard Hartley, and Ian Reid. Scalable deep k-subspace clustering. *arXiv preprint arXiv:1811.01045*, 2018.

[44] Teng Zhang, Arthur Szlam, and Gilad Lerman. Median k-flats for hybrid linear modeling with many outliers. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 234–241. IEEE, 2009.

[45] Teng Zhang, Arthur Szlam, Yi Wang, and Gilad Lerman. Hybrid linear modeling via local best-fit flats. *International journal of computer vision*, 100(3):217–240, 2012.