Information and Inference: A Journal of the IMA (2022) **00**, 1–28
https://doi.org/10.1093/imaiai/iaac008

Probabilistic methods for approximate archetypal analysis

Ruijian Han
Department of Statistics, The Chinese University of Hong Kong, Hong Kong, China

Braxton Osting
Department of Mathematics, University of Utah, Salt Lake City, UT, USA

Dong Wang
School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen, China

and

Yiming Xu
Department of Mathematics, University of Utah, Salt Lake City, UT, USA
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA
†Corresponding author. Email: yxu@math.utah.edu

[Received on 16 August 2021; revised on 11 February 2022; accepted on 16 March 2022]

Archetypal analysis (AA) is an unsupervised learning method for exploratory data analysis. One major challenge that limits the applicability of AA in practice is the inherent computational complexity of the existing algorithms. In this paper, we provide a novel approximation approach to partially address this issue. Utilizing probabilistic ideas from high-dimensional geometry, we introduce two preprocessing techniques to reduce the dimension and representation cardinality of the data, respectively. We prove that provided data are approximately embedded in a low-dimensional linear subspace and the convex hull of the corresponding representations is well approximated by a polytope with a few vertices, our method can effectively reduce the scaling of AA. Moreover, the solution of the reduced problem is near-optimal in terms of prediction errors. Our approach can be combined with other acceleration techniques to further mitigate the intrinsic complexity of AA. We demonstrate the usefulness of our results by applying our method to summarize several moderately large-scale datasets.

Keywords: alternating minimization; approximate convex hulls; archetypal analysis; dimensionality reduction; random projections; randomized SVD. 2020 Math Subject Classification: 65F55; 68W20; 68W25; 68W40.

1. Introduction

Archetypal analysis (AA) is an unsupervised learning method introduced by Cutler and Breiman in 1994 [10]. For fixed $k \in \mathbb{N}$, the method finds a convex polytope with k vertices, referred to as *archetypes*, in the convex hull of the data that explains the most variation of the data. Equivalently, given $\{x_i\}_{i\in[N]} \subset \mathbb{R}^d$, AA can be formulated as the following optimization problem:

$$\min_{\boldsymbol{A}\in\mathbb{R}^{N\times k}_{\mathrm{cs}},\boldsymbol{B}\in\mathbb{R}^{k\times N}_{\mathrm{cs}}} \frac{1}{\sqrt{N}}\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}\boldsymbol{B}\|_F \qquad \boldsymbol{X}=[x_1,\cdots,x_N]\in\mathbb{R}^{d\times N}, \tag{1.1}$$

where F denotes the Frobenius norm, and 'cs' stands for column stochastic matrices, which are entry-wise nonnegative matrices with each column summing to 1. The normalization factor $1/\sqrt{N}$ is introduced for convenience later. To understand this formulation, note that the columns of $\boldsymbol{XA}$ are the expected archetypes, and the columns of $\boldsymbol{B}$ correspond to the projection coefficients of the columns of $\boldsymbol{X}$ to the convex hull of the archetypes. Consequently, the objective defined in (1.1) represents the (average) variation of the data that cannot be explained by the convex combinations of the archetypes.

AA is closely related to other unsupervised learning methods such as the k-means, principal component analysis (PCA) and nonnegative matrix factorization (NMF) [19, 22]. In fact, AA can be seen as an interpolation between the k-means and PCA; it has more geometry than the former, while it is more restrictive than the latter due to additional convexity constraints. This allows AA to produce more interpretable results in many applications, e.g. in evolutionary biology [38], meanwhile raising additional questions of increased computational complexity. Under suitable statistical assumptions, the consistency and convergence of AA are established [33].

Despite offering interpretable results, AA did not gain equal attention compared with its alternatives. One possible reason, as pointed out in [6], is due to the lack of efficient computational resources for applying AA to large-scale datasets, which are becoming increasingly ubiquitous in the big-data era. Indeed, the optimization defined in (1.1) is non-convex, and one common approach to solving (1.1) is based on an alternating minimization algorithm [10], which will be reviewed in Section 2. The subproblems in the alternating minimization scheme are equivalent to quadratic programming problems (see Section 2), which makes the full loop for solving AA computationally intensive for moderately large dimension d and cardinality N.

The scope of this paper is to provide a promising perspective for addressing the theoretical computational challenges encountered in solving AA. Instead of focusing on optimizing the subproblem solvers to accelerate computation, we introduce two separate reduction techniques to downsize the problem before applying optimization methods to solve (1.1). We show that under appropriate conditions, a solution of the reduced AA (i) well-approximates the solution of the original problem (1.1) in terms of prediction error and (ii) can be obtained significantly faster than the original solution. Our approach relies on a few fundamental results in high-dimensional geometry. Note that our proposed method is a data preprocessing procedure by nature and complements the many existing methods to further accelerate computation.

1.1 *Related work*

Making AA practical for large-scale data analysis has been an active area of research in recent years. Various approaches have been proposed to attack the problem from different perspectives. For example, feasible optimization techniques such as projected gradients [31], active-subsets [6] and the Frank-Wolfe method [4] are considered for accelerating solving the quadratic programming problem in the alternating minimization scheme. Relaxation methods including decoupling [30] and sparse projections [1] are concerned with relaxing the alternating minimization into problems that enjoy better scalability properties. Another direction of work is centered around approximately solving AA by first reducing the cardinality of the data via sparse representation [27, 40]. Although these approaches are demonstrated to work well empirically, they either do not address the intrinsic complexity of the problem or lack theoretical guarantee on the quality of approximation. In the recent work [28], the authors proposed to use the coreset of the data to reduce the computational complexity of the objective function and theoretically quantified the approximation error.

Using approximate isometric embedding to reduce dimensionality is a fruitful idea in data analysis. The technique has been successfully applied to a variety of problems including least-squares regression [2, 12], clustering [5, 8, 29], low-rank approximation [7, 18, 41], NMF [14, 34] and tensor decomposition [3, 44].

1.2 *Contributions of this paper*

This paper proposes two novel reduction techniques which can be combined with existing approaches to mitigate the inherent complexity of AA. Both techniques come with theoretical guarantees on their approximation accuracy. In particular,

- We introduce a data compression technique based on a randomized Krylov subspace method [32] to reduce data dimension. This procedure allows us to circumvent frequent queries to high-dimensional data and is new in the context of AA.
- We propose to use random projections to compute an approximate convex hull of the data to reduce the cardinality of the dictionary to represent archetypes.
- We theoretically analyze the approximation accuracy and time complexity for both techniques. In particular, we show that the reduced AA gives a near-optimal solution but has significantly reduced complexity provided that the data are low-dimensional and approximately described by a few extreme patterns.

Our results yield an approximate algorithm that is capable of dealing with data that are large both in size and dimension. Numerical experiments are provided, which support and illustrate our theoretical findings.

1.3 *Outline*

The rest of the paper is organized as follows. In Section 2, we review the standard alternating minimization algorithm for solving AA as well as the corresponding computational challenges. In Section 3 and 4, we introduce two separate randomized techniques to reduce the data dimension and representation cardinality of the archetypes, respectively. We also quantify the approximation accuracy and the computational complexity for both techniques. In Section 5, we combine the ideas in Section 3 and 4 to devise an approximate algorithm for AA. We show that the proposed algorithm gives a near-optimal solution meanwhile having significantly reduced computational complexity for datasets that are approximately embedded in a low-dimensional subspace and well summarized via a few extreme points. We numerically verify our results in Section 6.

1.4 *Notation*

In the rest of the paper, $\boldsymbol{X} \in \mathbb{R}^{d\times N}$ denotes the data matrix. We always use $(\boldsymbol{A}_\star, \boldsymbol{B}_\star)$ to denote a minimizer to (1.1) and $\mathrm{opt}(\boldsymbol{X}) = \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F/\sqrt{N}$ the corresponding optimum value.

Denote $[m] = \{1, \ldots, m\} \subset \mathbb{N}$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, we denote by $\sigma_i(\boldsymbol{A})$ the ith largest singular value of $\boldsymbol{A}$, and $\boldsymbol{A}^\dagger$ the Moore–Penrose pseudoinverse of $\boldsymbol{A}$. For $T_1 \subset [m]$ and $T_2 \subset [n]$, we use notation $\boldsymbol{A}[T_1, :]$, $\boldsymbol{A}[-T_1, :]$, $\boldsymbol{A}[:, T_2]$ and $\boldsymbol{A}[:, -T_2]$ to denote the submatrices formed by taking the rows of $\boldsymbol{A}$ with indices in T_1, the rows of $\boldsymbol{A}$ with indices in $[m] \setminus T_1$, the columns of $\boldsymbol{A}$ with indices in T_2 and the columns of $\boldsymbol{A}$ with indices in $[n] \setminus T_2$, respectively. When talking about subspace embedding for $\boldsymbol{A}$, we

Algorithm 1: Alternating Minimization Algorithm for AA [10]

Input: $\{x_i\}_{i\in[N]}$: dataset, k: number of archetypes
Output: $\boldsymbol{A}$, $\boldsymbol{B}$
1: Initialize $\boldsymbol{XA}$
2: **while** not converged **do**
3: $\quad \boldsymbol{B} \leftarrow \arg\min_{\boldsymbol{B}'\in\mathbb{R}_{cs}^{k\times N}} \|\boldsymbol{X}-\boldsymbol{XAB}'\|_F^2$
4: $\quad \boldsymbol{A} \leftarrow \arg\min_{\boldsymbol{A}'\in\mathbb{R}_{cs}^{N\times k}} \|\boldsymbol{X}-\boldsymbol{XA}'\boldsymbol{B}\|_F^2$
5: **end while**
6: final update for $\boldsymbol{B}$: $\boldsymbol{B} \leftarrow \arg\min_{\boldsymbol{B}'\in\mathbb{R}_{cs}^{k\times N}} \|\boldsymbol{X}-\boldsymbol{XAB}'\|_F^2$
7: return $\boldsymbol{A}, \boldsymbol{B}$

view $\boldsymbol{A}$ as n points $\boldsymbol{A}[:,1],\cdots,\boldsymbol{A}[:,n]$ in the column space of $\boldsymbol{A}$, i.e. $\mathrm{col}(\boldsymbol{A})$. We use $\mathrm{conv}(\boldsymbol{A})$ and $\mathrm{ex}(\boldsymbol{A})$ to represent the convex hull of the columns of $\boldsymbol{A}$ and the corresponding extreme points, respectively.

Moreover, $\mathcal{O}(\cdot)$, $a(n_1,\cdots,n_\ell) \lesssim b(n_1,\cdots,n_\ell)$ and $a(n_1,\cdots,n_\ell) \gtrsim b(n_1,\cdots,n_\ell)$ are standard notation in complexity theory, where the implicit constants do not depend on the indices $n_1,\cdots,n_\ell$.

2. An alternating minimization algorithm for archetypal analysis

In this section, we review an alternating minimization algorithm for solving AA, due to Cutler and Breiman [10].

Note that (1.1) is a non-convex optimization. However, when fixing $\boldsymbol{A}$ or $\boldsymbol{B}$ and solving for the other, the problem becomes convex. This observation gives rise to the following alternating minimization algorithm for computing a stationary solution for (1.1).

The loop in Algorithm 1 updates $\boldsymbol{B}$ and $\boldsymbol{A}$ alternatingly. To analyze the computational complexity of these subroutines, we formulate the optimization problems in steps 3 and 4 more explicitly as follows.

In step 3, $\boldsymbol{A}$ is fixed and $\boldsymbol{B}$ needs to be updated. If we let $\boldsymbol{Z}=\boldsymbol{XA}$, then the optimization is equivalent to computing the projection coefficients for each column in $\boldsymbol{X}$ to $\mathrm{conv}(\boldsymbol{Z})$. In particular, we need to solve N independent k-dimensional quadratic programming problems

$$\min_{b\in\mathbb{R}^k,\|b\|_1=1,b\geq 0} \|\boldsymbol{Z}b-\boldsymbol{X}[:,i]\|_2^2 \qquad\qquad i\in[N].$$

In step 4, $\boldsymbol{B}$ is fixed and $\boldsymbol{A}$, or equivalently, $\boldsymbol{Z}$, needs to be updated. Using the Pythagorean theorem, one can first compute the least-squares solutions

$$\arg\min_{\boldsymbol{Z}\in\mathbb{R}^{d\times k}} \|\boldsymbol{X}-\boldsymbol{ZB}\|_F^2 = ((\boldsymbol{B}^T)^\dagger \boldsymbol{X}^T)^T = \boldsymbol{XB}^T(\boldsymbol{BB}^T)^{-1},$$

then update each column of $\boldsymbol{A}$ by taking projection

$$\min_{a\in\mathbb{R}^N,\|a\|_1=1,a\geq 0} \left\|\boldsymbol{X}a-\boldsymbol{XB}^T(\boldsymbol{BB}^T)^{-1}[:,i]\right\|_2^2 \qquad\qquad i\in[k]. \tag{2.1}$$

To accelerate computation, rather than updating $\boldsymbol{A}$ at once, one can use a Gauss–Seidel approach to update the columns of $\boldsymbol{A}$ sequentially [33]

$$\min_{a\in\mathbb{R}^N,\|a\|_1=1,a\geq 0}\left\|\boldsymbol{X}a-\frac{\boldsymbol{D}_i(\boldsymbol{B}[i,:])^T}{\|\boldsymbol{B}[i,:]\|_2^2}\right\|_2^2 \qquad i\in[k], \tag{2.2}$$

where $\boldsymbol{D}_i=\boldsymbol{X}-\boldsymbol{Z}[:,-i]\boldsymbol{B}[-i,:]$. The details of deriving (2.2) are given in Appendix A.

Either (2.1) or (2.2) involves solving k quadratic programming problems with variable dimension N. In the rest of the paper, we will use (2.2) in place of step 4 in Algorithm 1 to update $\boldsymbol{A}$.

For small k and large N, the computation time in step 3 scales linearly in N (assuming solving a k-dimensional quadratic programming problem takes constant time). For step 4, the computation time is approximately equal to a multiplicative constant (k) times the complexity of solving an N-dimensional quadratic programming problem, which can be computationally infeasible for large N. We will provide an accelerated approximate scheme for step 4 in Section 4. Moreover, when d is large, taking repeated numerical operations on $\boldsymbol{X}$ is inconvenient. We will introduce a data dimensionality reduction technique to address this issue in Section 3.

3. Data dimensionality reduction

We first consider the scenario where the data dimension is large. This may happen, for instance, when each data point is obtained from the discretization of a continuous function (e.g. time series) or encodes a high-resolution image. In this case, directly working with the data is inconvenient. Instead, we can embed $\boldsymbol{X}$ in a lower dimensional space while maintaining the convexity structure of $\boldsymbol{X}$. This compression will save us from frequently querying the columns of $\boldsymbol{X}$ in the iterative process for solving (1.1), which can be computationally expensive. A straightforward idea for embedding is via singular value decomposition (SVD), which we recall below

Definition 3.1. Suppose $\operatorname{rank}(\boldsymbol{X})=r\leq\min\{N,d\}$. The SVD of $\boldsymbol{X}$ is given by $\boldsymbol{X}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U}\in\mathbb{R}^{d\times r},\boldsymbol{V}\in\mathbb{R}^{r\times N}$ are the left and right singular vector matrices, respectively, and $\boldsymbol{\Sigma}\in\mathbb{R}^{r\times r}$ is a diagonal matrix with diagonal entries arranged in non-increasing order.

Under the columns of $\boldsymbol{U}$, $\boldsymbol{\Sigma}\boldsymbol{V}^T\in\mathbb{R}^{r\times N}$ provides a sparse representation for $\boldsymbol{X}$ (since $r\leq d$). If we first embed $\boldsymbol{X}$ in $\boldsymbol{U}$ using SVD and apply AA to $\boldsymbol{\Sigma}\boldsymbol{V}^T$, then for every feasible $(\boldsymbol{A},\boldsymbol{B})$, by the unitary invariance of Frobenius norm,

$$\left\|\boldsymbol{\Sigma}\boldsymbol{V}^T-\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{B}\right\|_F^2=\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}\boldsymbol{B}\|_F^2, \tag{3.1}$$

which establishes the equivalence between (1.1) and the AA under the SVD representation.

If $\boldsymbol{X}$ has full rank but possesses low-rank structure, one may use a truncated SVD to further reduce the data dimension at a minor cost of accuracy, as made precise in the following theorem:

Theorem 3.2. Suppose $p\leq r=\operatorname{rank}(\boldsymbol{X})$. Denote by $\boldsymbol{U}_p,\boldsymbol{V}_p$ the first p columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively, and $\boldsymbol{\Sigma}_p$ the top $p\times p$ submatrix of $\boldsymbol{\Sigma}$. Let $(\widetilde{\boldsymbol{A}},\widetilde{\boldsymbol{B}})$ be a solution to the AA for the truncated

SVD representation of $\boldsymbol{X}$ at pth level

$$\min_{\boldsymbol{A}\in\mathbb{R}_{\mathrm{cs}}^{N\times k},\boldsymbol{B}\in\mathbb{R}_{\mathrm{cs}}^{k\times N}} \frac{1}{\sqrt{N}} \left\| \boldsymbol{\Sigma}_p \boldsymbol{V}_p^T - \boldsymbol{\Sigma}_p \boldsymbol{V}_p^T \boldsymbol{A}\boldsymbol{B} \right\|_F.$$

Then,

$$\frac{1}{\sqrt{N}} \left\| \boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}} \right\|_F \le \mathrm{opt}(\boldsymbol{X}) + 4\sigma_{p+1}(\boldsymbol{X}). \tag{3.2}$$

Proof. Let

$$\boldsymbol{X}_p := \boldsymbol{U}_p \boldsymbol{\Sigma}_p \mathbf{V}_p \qquad\qquad \boldsymbol{X}_{-p} := \boldsymbol{X} - \boldsymbol{X}_p. \tag{3.3}$$

By the Eckart–Young theorem [13], $\boldsymbol{X}_p$ is the best rank-p approximation for $\boldsymbol{X}$ in the spectral norm, with approximation error $\|\boldsymbol{X}_{-p}\|_2 = \sigma_{p+1}(\boldsymbol{X})$. Let $(\boldsymbol{A}_\star, \boldsymbol{B}_\star)$ be a solution to (1.1). Consequently,

$$\begin{aligned}
\|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F &\le \|\boldsymbol{X}_p - \boldsymbol{X}_p\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F + \|\boldsymbol{X}_{-p} - \boldsymbol{X}_{-p}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \\
&\le \|\boldsymbol{X}_p - \boldsymbol{X}_p\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F + \|\boldsymbol{X}_{-p} - \boldsymbol{X}_{-p}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \\
&\le \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F + \|\boldsymbol{X}_{-p} - \boldsymbol{X}_{-p}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F + \|\boldsymbol{X}_{-p} - \boldsymbol{X}_{-p}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \\
&\le \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F + 2\|\boldsymbol{X}_{-p}\|_F + \|\boldsymbol{X}_{-p}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F + \|\boldsymbol{X}_{-p}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F.
\end{aligned} \tag{3.4}$$

Since $\boldsymbol{A}_\star, \widetilde{\boldsymbol{A}}, \boldsymbol{B}_\star, \widetilde{\boldsymbol{B}}$ are column stochastic matrices, so are $\boldsymbol{A}_\star\boldsymbol{B}_\star$ and $\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}$. It follows from direct computation and Cauchy–Schwarz inequality that

$$\begin{aligned}
\|\boldsymbol{X}_{-p}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F = \sqrt{\sum_{i\in[N]} \|\boldsymbol{X}_{-p}(\boldsymbol{A}_\star\boldsymbol{B}_\star)[:,i]\|_2^2} &\le \sqrt{\sum_{i\in[N]} \|\boldsymbol{X}_{-p}\|_2^2 \|(\boldsymbol{A}_\star\boldsymbol{B}_\star)[:,i]\|_2^2} \\
&\le \sqrt{\sum_{i\in[N]} \|\boldsymbol{X}_{-p}\|_2^2 \|(\boldsymbol{A}_\star\boldsymbol{B}_\star)[:,i]\|_1^2} \\
&= \|\boldsymbol{X}_{-p}\|_2\sqrt{N}.
\end{aligned} \tag{3.5}$$

Similarly,

$$\|\boldsymbol{X}_{-p}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \le \|\boldsymbol{X}_{-p}\|_2\sqrt{N}. \tag{3.6}$$

Plugging (3.5) and (3.6) into (3.4) and dividing by $\sqrt{N}$ yields

$$\begin{aligned}
\frac{1}{\sqrt{N}}\|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \le \mathrm{opt}(\boldsymbol{X}) + \frac{2}{\sqrt{N}}\|\boldsymbol{X}_{-p}\|_F + 2\|\boldsymbol{X}_{-p}\|_2 &\le \mathrm{opt}(\boldsymbol{X}) + 4\|\boldsymbol{X}_{-p}\|_2 \\
&= \mathrm{opt}(\boldsymbol{X}) + 4\sigma_{p+1}(\boldsymbol{X}),
\end{aligned}$$

completing the proof. □

Thus, for data $\boldsymbol{X}$ that admits a good low-rank approximation, AA applied to the truncated SVD representation yields a near-optimal solution in terms of prediction errors. In this case, the data dimension can be significantly reduced to streamline computation. However, to obtain truncated SVD representations, one often needs to compute the full SVD of $\boldsymbol{X}$, which has complexity $\mathcal{O}(dN \min\{d, N\})$. For large d and N, this procedure is computationally intensive. To address this issue, we consider an approximate version of the best rank-p approximation without taking the SVD of $\boldsymbol{X}$.

DEFINITION 3.3. A matrix $\widetilde{\boldsymbol{X}}_p$ is a $(1+\varepsilon)$ rank-p approximation to $\boldsymbol{X}$ if $\operatorname{rank}(\widetilde{\boldsymbol{X}}_p) \le p$ and

$$\|\boldsymbol{X} - \widetilde{\boldsymbol{X}}_p\|_2 \le (1+\varepsilon)\|\boldsymbol{X} - \boldsymbol{X}_p\|_2, \tag{3.7}$$

where $\boldsymbol{X}_p$ is the best rank-p approximation to $\boldsymbol{X}$ as defined in (3.3).

Before turning to discuss how to find such an $\widetilde{\boldsymbol{X}}_p$, we consider a few consequences assuming its existence. Similar to the previous discussion, we can apply AA to $\widetilde{\boldsymbol{X}}_p$, which can be efficiently represented using the SVD. As will be seen shortly, computing the SVD of $\widetilde{\boldsymbol{X}}_p$ is much cheaper than $\boldsymbol{X}$ when p is small. On the other hand, let $\widetilde{\boldsymbol{X}}_p = \widetilde{\boldsymbol{U}}_p \widetilde{\boldsymbol{\Sigma}}_p \widetilde{\boldsymbol{V}}_p^T$ be the SVD of $\widetilde{\boldsymbol{X}}_p$ and define

$$\widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{\Sigma}}_p \widetilde{\boldsymbol{V}}_p^T \qquad \Longrightarrow \qquad \widetilde{\boldsymbol{X}}_p = \widetilde{\boldsymbol{U}}_p \widetilde{\boldsymbol{X}}. \tag{3.8}$$

The following theorem quantifies the approximation error if we use $\widetilde{\boldsymbol{X}}$ in place of $\boldsymbol{X}$ for AA:

THEOREM 3.4. Let $\varepsilon > 0$. Suppose $\widetilde{\boldsymbol{X}}_p$ is a $(1+\varepsilon)$ rank-p approximation to $\boldsymbol{X}$, and $\widetilde{\boldsymbol{X}}$ is the representation of $\widetilde{\boldsymbol{X}}_p$ under the left singular vectors. Let $(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}})$ be a solution to the AA applied to $\widetilde{\boldsymbol{X}}$

$$\min_{\boldsymbol{A} \in \mathbb{R}_{\mathrm{cs}}^{N \times k}, \boldsymbol{B} \in \mathbb{R}_{\mathrm{cs}}^{k \times N}} \frac{1}{\sqrt{N}} \left\| \widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}} \boldsymbol{A} \boldsymbol{B} \right\|_F. \tag{3.9}$$

hen,

$$\frac{1}{\sqrt{N}} \|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \le \operatorname{opt}(\boldsymbol{X}) + 4(1+\varepsilon)\sigma_{p+1}(\boldsymbol{X}).$$

Proof. Proceeding similarly as proof of Theorem 3.2 with $\boldsymbol{X}_p, \boldsymbol{X}_{-p}$ replaced by $\widetilde{\boldsymbol{X}}_p$ and $\widetilde{\boldsymbol{X}}_{-p} = \boldsymbol{X} - \widetilde{\boldsymbol{X}}_p$, respectively,

$$\frac{1}{\sqrt{N}} \|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \le \operatorname{opt}(\boldsymbol{X}) + 4\|\widetilde{\boldsymbol{X}}_{-p}\|_2^{(3.7)} \le \operatorname{opt}(\boldsymbol{X}) + 4(1+\varepsilon)\sigma_{p+1}(\boldsymbol{X}).$$

□

Theorem 3.4 implies that for $\boldsymbol{X}$ with small best rank-p approximation error, using the SVD representation of $\widetilde{\boldsymbol{X}}_p$ will only result in a small impact on prediction accuracy. The following algorithm, due to Musco and Musco [32], provides a way to compute $\widetilde{\boldsymbol{X}}_p$ (i.e. $\widetilde{\boldsymbol{X}}$) via randomized block Krylov methods. The details of the algorithm are given in Algorithm 2.

Algorithm 2: Block Krylov Iteration [32]

Input: $\boldsymbol{X} \in \mathbb{R}^{d\times N}$: data matrix, p: approximation rank, s: power parameter
Output: $\widetilde{\boldsymbol{X}}_p$

1: generate p (Gaussian) random initializations: $\boldsymbol{S} \in \mathbb{R}^{N\times p}$, $\boldsymbol{S}_{ij} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,1)$
2: construct the Krylov subspace: $\boldsymbol{K} = [\boldsymbol{XS}, (\boldsymbol{XX}^T)\boldsymbol{XS}, \cdots, (\boldsymbol{XX}^T)^{s-1}\boldsymbol{XS}] \in \mathbb{R}^{d\times(sp)}$
3: compute the QR decomposition for $\boldsymbol{K}$: $\boldsymbol{K} = \boldsymbol{QR}$
4: compute the SVD for $\boldsymbol{X}_{\text{emd}} = \boldsymbol{X}^T\boldsymbol{Q}$: $\boldsymbol{X}_{\text{emd}} = \boldsymbol{U}_{\text{emd}}\Sigma_{\text{emd}}\boldsymbol{V}_{\text{emd}}^T$
5: compute $\widetilde{\boldsymbol{X}}$: $\widetilde{\boldsymbol{X}}_p = \boldsymbol{LL}^T\boldsymbol{X}$, with $\boldsymbol{L} = \boldsymbol{QV}_{\text{emd}}[:, 1:p]$

The following lemma says that, with high probability, $\widetilde{\boldsymbol{X}}_p$ returned by Algorithm 2 is a good approximation to $\boldsymbol{X}_p$:

For $\varepsilon, \delta > 0$, there exist absolute constants $C_1, C_2 > 0$ such that if

$$s > \frac{C_1}{\sqrt{\varepsilon}} \log\left(\frac{N}{\varepsilon\delta}\right) \qquad\qquad p \geq C_2 \log\left(\frac{4}{\delta}\right), \tag{3.10}$$

hen for with probability at least $1-\delta$, the $\widetilde{\boldsymbol{X}}_p$ in Algorithm 2 satisfies (3.7).

Proof. Lemma 3.1 is a probabilistic version of ([32], Theorem 1) where a fixed probability (0.99) is used instead of $1-\delta$ for an arbitrary δ. Nevertheless, the proof is the similar except one needs to apply sharp concentration inequalities to bound extreme singular values of Gaussian matrices ([43], Corollary 7.3.3), ([36], Theorem 1.2) to control the failure probability. □

REMARK 3.1. Other randomized low-rank approximation algorithms may also be used in place of Algorithm 2. For example, one can use the randomized simultaneous iteration to compute $\widetilde{\boldsymbol{X}}_p$ [18; 45]. Under the same approximation error ε and failure probability δ, the sample complexity of this method has a slightly worse dependence on ε (i.e. $s = \mathcal{O}(\log(N/\varepsilon\delta)/\varepsilon)$) than (3.10). As such theoretical discrepancy was also manifested in several empirical studies in [32], we use Algorithm 2 to compute $\widetilde{\boldsymbol{X}}_p$ in this article.

REMARK 3.2. The desired low-rank approximation $\widetilde{\boldsymbol{X}}_p$ is computed under the spectral norm, which is necessary in the derivation of approximation error in Theorem 3.4. Other randomized algorithms based on oblivious sketching [8; 37; 45] or leverage score sampling [9] only produce low-rank approximations under the Frobenius norm. Since an error bound under the Frobenius norm does not imply a similar bound under the spectral norm, these methods do not directly work for the problem considered in this paper.

To apply Theorem 3.4, we need to compute the SVD representation of the low-rank approximation matrix $\widetilde{\boldsymbol{X}}_p$, that is $\widetilde{\boldsymbol{X}}$, rather than $\widetilde{\boldsymbol{X}}_p$ itself; see (3.8). Since the output of Algorithm 2 is the full low-rank matrix $\widetilde{\boldsymbol{X}}_p$, finding its SVD representation may incur additional computational cost for our purpose. However, in Algorithm 2, $\widetilde{\boldsymbol{X}}$ can be read off the shelf as

$$\widetilde{\boldsymbol{X}} = \boldsymbol{\Sigma}_{\text{emd}}[1:p, 1:p](\boldsymbol{U}_{\text{emd}}[:, 1:p])^T, \tag{3.11}$$

where $\boldsymbol{\Sigma}_{\text{emd}}$ and $\boldsymbol{U}_{\text{emd}}$ are computed in step 4. Thus, the total cost for $\widetilde{\boldsymbol{X}}_p$ is the computational complexity for the first four steps in Algorithm 2.

THEOREM 3.5. Let $\widetilde{\boldsymbol{X}}$ be the SVD representation of $\widetilde{\boldsymbol{X}_p}$ in Algorithm 2 as in (3.11). Then, the computational complexity for $\widetilde{\boldsymbol{X}}$ is $\mathcal{O}(dNps + dp^2s^2 + Nps\min\{N, ps\})$.

Proof. Step 1 in Algorithm 2 generates a random Gaussian matrix which takes time $\mathcal{O}(Np)$. Step 2 computes the Krylov subspace basis which takes time $\mathcal{O}(dNps)$. The QR decomposition of $\boldsymbol{K}$ in step 3 takes time $\mathcal{O}(dp^2s^2)$. In step 4, we first compute $\boldsymbol{X}_{\text{emd}}$, which takes time $\mathcal{O}(dNps)$, then compute the SVD of $\boldsymbol{X}_{\text{emd}}$, which takes time $\mathcal{O}(Nps\cdot\min\{N, ps\})$. Computing $\widetilde{\boldsymbol{X}} = \boldsymbol{\Sigma}_{\text{emd}}[1:p, 1:p](\boldsymbol{U}_{\text{emd}}[:, 1:p])^T$ takes time $\mathcal{O}(Np)$. □

REMARK 3.3. When $ps \ll \min\{d, N\}$, the computational complexity of $\widetilde{\boldsymbol{X}}$ becomes $\mathcal{O}(dNps)$, which is significantly smaller than $\mathcal{O}(dN\min\{d, N\})$.

Setting $\varepsilon = 1$ in Lemma 3.1 and combining Theorem 3.4, we have the following result:

THEOREM 3.6. Let $\widetilde{\boldsymbol{X}}$ be the SVD representation of $\widetilde{\boldsymbol{X}}_p$ returned by Algorithm 2. Let $(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}})$ be a solution to the AA for $\widetilde{\boldsymbol{X}}$

$$\min_{\boldsymbol{A}\in\mathbb{R}_{\text{cs}}^{N\times k}, \boldsymbol{B}\in\mathbb{R}_{\text{cs}}^{k\times N}} \frac{1}{\sqrt{N}}\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\boldsymbol{A}\boldsymbol{B}\|_F. \tag{3.12}$$

For $\delta > 0$, if p satisfies (3.10) and $s > C\log(N/\delta)$ for some absolute constant $C > 0$, then with probability at least $1 - \delta$,

$$\frac{1}{\sqrt{N}}\|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \le \text{opt}(\boldsymbol{X}) + 8\sigma_{p+1}(\boldsymbol{X}).$$

REMARK 3.4. Fixing δ small, say $\delta = 0.01$, $s = \mathcal{O}(\log N)$. The computational complexity of $\widetilde{\boldsymbol{X}}$ is $\mathcal{O}(dNp\log N + dp^2\log^2 N + Np\log N\min\{N, p\log N\})$. Consequently, for data $\boldsymbol{X}$ that can be well approximated via low-rank matrices with approximation rank $p \ll N$, using Algorithm 2 can effectively reduce the dimension of AA.

4. Representation cardinality reduction

We now consider the situation where the dataset has a large cardinality. In this case, to reduce computational complexity, we propose to use a parsimonious subset of points in $\boldsymbol{X}$ to approximately represent $\text{conv}(\boldsymbol{X})$, i.e. we wish to find a small subset $T \subset [N]$ such that

$$\text{conv}(\boldsymbol{X}_T) \approx \text{conv}(\boldsymbol{X}), \tag{4.1}$$

where $\boldsymbol{X}_T := \boldsymbol{X}[:, T]$ and $\approx$ will be made rigorous later. We will refer to $\text{conv}(\boldsymbol{X}_T)$ as an *approximate convex hull* of $\boldsymbol{X}$.

The idea of using subsets of $\boldsymbol{X}$ (i.e. extreme points) to represent $\text{conv}(\boldsymbol{X})$ has been considered in [27, 40], where exact equality in (4.1) is expected. Here, we only ask for approximate representation of $\text{conv}(\boldsymbol{X})$ (allowing for a small approximation error), so that it is possible to further reduce the cardinality of the representation set for the archetypes (Fig. 1).

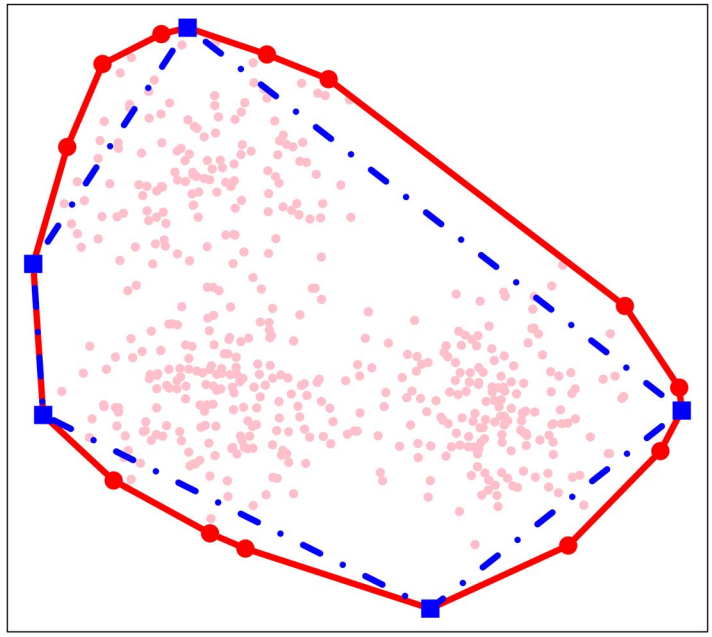

Figure 1. An example of the convex hull (red solid curves) and an approximate convex hull (blue dashed curve) of a randomly generated dataset.

Similar to the discussion in the previous section, we first give a few consequences assuming $\boldsymbol{X}_T$ exists.

DEFINITION 4.1. We say that $\boldsymbol{X}_T$ is an ε-approximate convex hull of $\boldsymbol{X}$ if

$$d_H(\text{conv}(\boldsymbol{X}_T), \text{conv}(\boldsymbol{X})) \leq \varepsilon, \tag{4.2}$$

where $d_H(X, Y) := \max\left\{\sup_{x\in X} d(x, Y), \sup_{y\in Y} d(X, y)\right\}$ is the Hausdorff distance.

THEOREM 4.2. For $\varepsilon > 0$ and $T \subseteq [N]$, suppose $\boldsymbol{X}_T$ is a $(\text{opt}(\boldsymbol{X}) \cdot \varepsilon)$-approximate convex hull of $\boldsymbol{X}$. Consider the following AA optimization problem constrained to $\text{conv}(\boldsymbol{X}_T)$:

$$\min_{\boldsymbol{A}\in\mathbb{R}_{\text{cs}}^{|T|\times k}, \boldsymbol{B}\in\mathbb{R}_{\text{cs}}^{k\times N}} \frac{1}{\sqrt{N}}\|\boldsymbol{X} - \boldsymbol{X}_T\boldsymbol{A}\boldsymbol{B}\|_F. \tag{4.3}$$

Then, the archetype points given by the solution of (4.3) provide a $(1+\varepsilon)$-approximation to the solution for (1.1) in terms of prediction errors

$$\min_{\boldsymbol{A}\in\mathbb{R}_{\text{cs}}^{|T|\times k}, \boldsymbol{B}\in\mathbb{R}_{\text{cs}}^{k\times N}} \frac{1}{\sqrt{N}}\|\boldsymbol{X} - \boldsymbol{X}_T\boldsymbol{A}\boldsymbol{B}\|_F \leq (1+\varepsilon)\text{opt}(\boldsymbol{X}).$$

Proof. For an optimal solution $(\boldsymbol{A}_\star, \boldsymbol{B}_\star)$ of (1.1) that resides on the boundary of $\mathrm{conv}(\boldsymbol{X})$ (such a solution always exists [10]), consider the projection of each column of $\boldsymbol{X}\boldsymbol{A}_\star$ to $\mathrm{conv}(\boldsymbol{X}_T)$, and denote the projected points as $\boldsymbol{Z}$. Note that $\boldsymbol{Z}$ is well-defined as $\mathrm{conv}(\boldsymbol{X}_T) \subset \mathrm{conv}(\boldsymbol{X})$. By the triangle inequality, the distance between each column of $\boldsymbol{X}$ and $\mathrm{conv}(\boldsymbol{Z})$ is bounded by the sum of the distance between the column of $\boldsymbol{X}$ and $\mathrm{conv}(\boldsymbol{X}\boldsymbol{A}_\star)$ and $d_H(\boldsymbol{X}\boldsymbol{A}_\star, \boldsymbol{Z})$. Since $\boldsymbol{X}_T$ gives an $(\mathrm{opt}(\boldsymbol{X}) \cdot \varepsilon)$-approximate convex hull of $\boldsymbol{X}$,

$$d_H(\boldsymbol{X}\boldsymbol{A}_\star, \boldsymbol{Z}) \le d_H(\mathrm{conv}(\boldsymbol{X}), \mathrm{conv}(\boldsymbol{X}_T)) \le \varepsilon \cdot \mathrm{opt}(\boldsymbol{X}) = \frac{\varepsilon}{\sqrt{N}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F. \tag{4.4}$$

t follows from direct computation that

$$\begin{aligned}
\min_{\boldsymbol{A}\in\mathbb{R}_{\mathrm{cs}}^{|T|\times k}, \boldsymbol{B}\in\mathbb{R}_{\mathrm{cs}}^{k\times N}} \frac{1}{N}\|\boldsymbol{X} - \boldsymbol{X}_T\boldsymbol{A}\boldsymbol{B}\|_F^2 &\le \min_{\boldsymbol{B}\in\mathbb{R}_{\mathrm{cs}}^{k\times N}} \frac{1}{N}\|\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{B}\|_F^2 \\
&= \frac{1}{N}\sum_{i\in[N]} d(x_i, \mathrm{conv}(\boldsymbol{Z}))^2 \\
&\le \frac{1}{N}\sum_{i\in[N]} (d(x_i, \mathrm{conv}(\boldsymbol{X}\boldsymbol{A}_\star)) + d_H(\boldsymbol{X}\boldsymbol{A}_\star, \boldsymbol{Z}))^2 \\
&\le (1+\varepsilon)^2 \cdot \frac{1}{N}\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F^2,
\end{aligned}$$

where the last inequality follows from (4.4) and Cauchy–Schwarz inequality. Taking the square root on both sides completes the proof. □

Theorem 4.2 establishes an approximate equivalence between the solutions of (4.3) and (1.1) in terms of objective values. Compared with (1.1), the dimension of $\boldsymbol{A}$ is significantly reduced provided $|T| \ll N$, while the dimension of $\boldsymbol{B}$ stays unchanged.

To see the computational gain brought by working with (4.3), recall the alternating minimization in Section 2. When $\boldsymbol{B}$ is fixed and $\boldsymbol{A}$ is updated, one needs to solve k quadratic programming problems with variable dimensions equal to the number of rows of $\boldsymbol{A}$. For certain optimization methods such as the ellipsoid method, the complexity of quadratic programming problems with positive-definite quadratic matrix has weakly polynomial time (of the variable dimension) [24]. Thus, when $|T| \ll N$, a notable acceleration is expected for the subroutine of updating $\boldsymbol{A}$, which justifies the significance of using a parsimonious subset of points to represent the archetypes.

On the other hand, when $\boldsymbol{A}$ is fixed and $\boldsymbol{B}$ is updated, one needs to compute the projection of each column of $\boldsymbol{X}$ to $\mathrm{conv}(\boldsymbol{X}\boldsymbol{A})$. This step is the same in both (1.1) and (4.3) and consists of N-independent quadratic programming problems with variable dimension k. In this case, it is possible to take an additional step of acceleration via parallelization combined with the coreset approximation [28], which reduces the computation of N projection coefficient vectors to a small subset of points in $\boldsymbol{X}$ with appropriate weights, similar to the ideas of quadrature. Indeed, given a coreset $\boldsymbol{X}_C \subset \boldsymbol{X}$ and appropriate weight diagonal matrix $\boldsymbol{W}$, one can approximate the objective function in (1.1) with $\|\boldsymbol{W}\boldsymbol{X}_C - \boldsymbol{W}\boldsymbol{X}\boldsymbol{A}\boldsymbol{B}\|_F/\sqrt{N}$. Note the complexity of the subproblem for updating $\boldsymbol{B}$ in the alternating minimization algorithm is proportional to the number of points in the objective function. Therefore, when $|\boldsymbol{X}_C| \ll |\boldsymbol{X}|$, the step of solving the $\boldsymbol{B}$-subproblem can be significantly accelerated. Combining the idea of coreset with (4.4) yields an approximate objective function $\|\boldsymbol{W}\boldsymbol{X}_C - \boldsymbol{W}\boldsymbol{X}_T\boldsymbol{A}\boldsymbol{B}\|_F/\sqrt{N}$, which

has significantly reduced complexity when solved by the alternating minimization algorithm. The details are not discussed here.

We next discuss how to find a 'small' subset T such that (4.2) is satisfied. Note that to represent $\text{conv}(\boldsymbol{X})$, it suffices to consider the extreme points of $\boldsymbol{X}$. In other words, we will find a subset $\boldsymbol{X}_T \subset \text{ex}(\boldsymbol{X})$ whose convex hull can well approximate $\text{conv}(\boldsymbol{X})$. As will be seen below, this procedure can be effectively implemented by taking random projections. Indeed, random projections are linear maps whose inverse image of the extreme points of a convex set are a subset of the extreme points of the inverse image of that convex set [25]. Similar ideas have been used in the empirical study of AA to seek extreme points [11, 40].

Finding all the extreme points of $\text{conv}(\boldsymbol{X})$ may itself be computationally demanding unless $\text{ex}(\boldsymbol{X})$ is small. When $\text{conv}(\boldsymbol{X})$ can be well approximately using a few extreme points, it is desired to single them out to further shrink the complexity of the problem at a 'small' sacrifice of accuracy. To this end, we need to know which extreme points are more important than the others in terms of composing $\text{conv}(\boldsymbol{X})$. The following result, which originally appeared in [17], is precisely what is needed here.

Observe that under a random projection $v \in \mathbb{S}^{d-1}$, the points in $\boldsymbol{X}$ have projected values $\{\langle x_i, v\rangle\}_{i\in[N]}$, which with probability one have a unique maximum. The inverse image of the maximum is an element in $\text{ex}(\boldsymbol{X})$. Thus, throwing away a null set, we can partition the unit sphere $\mathbb{S}^{d-1}$ as follows:

$$\mathbb{S}^{d-1} = \bigsqcup_{x\in\text{ex}(\boldsymbol{X})} V_x \qquad\qquad V_x = \{v \in \mathbb{S}^{d-1} : v^T x > v^T x_i, i \in [N]\}.$$

For $x \in \text{ex}(\boldsymbol{X})$, its curvature is defined as

$$\kappa(x) = \frac{|V_x|}{|\mathbb{S}^{d-1}|},$$

which is the relative area of the directions that distinguish x as the maximum to the unit sphere in $\mathbb{R}^d$. By definition, points with larger curvature are more likely to be sampled if v is uniformly drawn from $\mathbb{S}^{d-1}$; in fact, they are also more 'important' as specified by the following lemma ([17], Theorem 3.4):

LEMMA 4.1. Let $d \geq 2$ and $S \subset [N]$. Suppose that both $\text{conv}(\boldsymbol{X}_S)$ and $\text{conv}(\boldsymbol{X})$ are non-degenerate (i.e. with nonempty interior), and $R := \max_{i\in[N]} \|x_i\|_2$. Then,

$$d_H(\text{conv}(\boldsymbol{X}), \text{conv}(\boldsymbol{X}_S)) \leq \min\left\{\sqrt{2}\pi(2\omega)^{\frac{1}{d-1}}, 2\right\} \cdot R \qquad\qquad \omega = \sum_{i\in[N]\setminus S} \kappa(x_i). \tag{4.5}$$

As a result, to compute a sparse approximate convex hull, it suffices to use high-curvature points to approximately represent $\text{conv}(\boldsymbol{X})$. To find high-curvature points, we apply a Monte-Carlo (MC) procedure to estimate the curvature of each point and then truncate at some thresholding parameter. The details are given in Algorithm 3.

A similar MC method based on a different truncation rule has been proposed [17, Algorithm 1), where points are removed whenever their estimated curvatures are below some fixed threshold. To ensure that the remaining points have large cumulative curvature, this algorithm requires the thresholding parameter to be overly small, leaving most points unremoved. To facilitate parsimony, Algorithm 3 first sorts points based on their estimated curvatures, then truncates based on the estimated cumulative curvatures.

Algorithm 3: Approximate Convex Hull

Input: $\{x_i\}_{i\in[N]}$: dataset, M: number of projections, η: approximation accuracy
Output: approximate convex hull $\mathrm{conv}(\{x_i\}_{i\in T})$

1: $e_j = 0$ for $j \in [N]$.
2: **for** $i = 1, \cdots, M$ **do**
3: $\quad v_i \sim \mathrm{Uniform}(\mathbb{S}^{d-1})$
4: $\quad u_i = \arg\max_j v_i^T x_j$
5: $\quad e_{u_i} \leftarrow e_{u_i} + 1$
6: **end for**
7: sort $\{e_j\}_{j\in[N]}$ in decreasing order as $e^{(1)} \geqslant \cdots \geqslant e^{(N)}$
8: compute $L = \min\left\{\ell : \frac{1}{M}\sum_{j\in[\ell]} e^{(j)} > 1 - \eta/3\right\}$
9: compute $L \leftarrow \max\{L, d+1\}$
10: let T be the index set of $e^{(1)}, \cdots, e^{(L)}$ and return $\{x_j\}_{j\in T}$

The computational complexity of Algorithm 3 can be obtained from direct computation.

THEOREM 4.3. The computational complexity for Algorithm 3 is $\mathcal{O}(MNd + N\log N)$.

Proof. The MC procedure in Algorithm 3 (step 2–6) involves M repetitions of computing N d-dimensional vector inner product and finding the (index of the) maximum of the projected points, which takes time $\mathcal{O}(MNd)$ in total. Step 7 is a simple sorting that has complexity $\mathcal{O}(N\log N)$ using Merge sort [23]. □

We will show that for large M, with high probability, the output of Algorithm 3 satisfies (4.2) with $\varepsilon = \min\left\{\sqrt{2}\pi\eta^{\frac{1}{d-1}}, 2\right\}\cdot R$. Without loss of generality, in the following discussion, we assume $|\mathrm{ex}(\boldsymbol{X})| = h$ and

$$\kappa(x_1) \geq \kappa(x_2) \geq \cdots \geq \kappa(x_h) > \kappa(x_{h+1}) = \cdots = \kappa(x_N) = 0. \tag{4.6}$$

We have the following theorem:

THEOREM 4.4. Let T be the subset returned by Algorithm 3, and $R = \max_{i\in[N]} \|x_i\|_2$. Suppose $\mathrm{conv}(\boldsymbol{X}_D)$ is non-degenerate for every $D \subset [N]$ with $|D| > d$. Denote q as the smallest integer such that $\sum_{i\in[q]} \kappa(x_i) \geq 1 - \eta/18$, i.e.

$$q := \min\left\{j : \sum_{i\in[q]} \kappa(x_i) \geq 1 - \frac{\eta}{18}\right\}, \tag{4.7}$$

and the truncation gap

$$\Delta := \kappa(x_q) - \kappa(x_{q+1}) > 0.$$

If

$$M \geq \max\left\{\frac{324q^2}{\eta^2}, \frac{4}{\Delta^2}\right\} \log\left(\frac{3N}{\sqrt{\delta}}\right), \tag{4.8}$$

then with probability at least $1-\delta$, $|T| \leq \max\{q, d+1\}$ and

$$d_H(\mathrm{conv}(\boldsymbol{X}_T), \mathrm{conv}(\boldsymbol{X})) \leq \min\left\{\sqrt{2}\pi\,\eta^{\frac{1}{d-1}}, 2\right\} \cdot R. \tag{4.9}$$

REMARK 4.1. Setting the upper bound in (4.9) equal to $\mathrm{opt}(\boldsymbol{X})\varepsilon$ yields

$$M \geq \max\left\{324q^2 \left(\frac{2\pi^2 R^2}{\mathrm{opt}(\boldsymbol{X})^2\varepsilon^2}\right)^{d-1}, \frac{4}{\Delta^2}\right\} \log\left(\frac{3N}{\sqrt{\delta}}\right),$$

which has an unpleasant but expected exponential dependence on d (curse of dimensionality). For datasets with low-dimensional structure, i.e. well approximated via rank-p matrices with $p \ll d$, it is possible to use ideas in Section 3 to improve the exponential dimension dependence to p (Algorithm 4).

Proof of Theorem 4.4 Note that step 9 in Algorithm 3 ensures that $\mathrm{conv}(\boldsymbol{X}_T)$ is non-degenerate. Therefore, to show (4.9), by (4.5), it suffices to show $\sum_{i\in[N]\setminus T}\kappa(x_i) \leq \eta/2$, or equivalently, $\sum_{i\in T}\kappa(x_i) \geq 1-\eta/2$.

We first show that for M satisfying (4.8), with high probability, the estimated curvatures e_j/M are close to their expectations for all reasonably large e_j. Note for every $j \in [q]$, e_j is a sum of M-independent Bernoulli random variables with parameter $\kappa(x_j)$, and the tail sum $\sum_{j>q} e_j$ is a sum of M-independent Bernoulli random variables with parameter $\sum_{j>q}\kappa(x_j) < \eta/18$. Thus, by Hoeffding's inequality [20],

$$\mathbb{P}\left[\left|\frac{1}{M}e_j - \kappa(x_j)\right| \leq \frac{\eta}{18q}\right] \geq 1 - 2\exp\left(-\frac{M\eta^2}{162q^2}\right)$$

$$\mathbb{P}\left[\left|\frac{1}{M}\sum_{j>q} e_j - \sum_{j>q}\kappa(x_j)\right| \leq \frac{\eta}{18q}\right] \geq 1 - 2\exp\left(-\frac{M\eta^2}{162q^2}\right).$$

Taking a union bound over $j \in [q]$ and combining the two inequalities yields

$$\mathbb{P}\left[\left\{\max_{j\in[q]}\left|\frac{1}{M}e_j - \kappa(x_j)\right|, \left|\frac{1}{M}\sum_{j>q} e_j - \sum_{j>q}\kappa(x_j)\right|\right\} \leq \frac{\eta}{18q}\right]$$

$$\geq 1 - 2(q+1)\exp\left(-\frac{M\eta^2}{162q^2}\right) \geq 1 - 4q\exp\left(-\frac{M\eta^2}{162q^2}\right). \tag{4.10}$$

The right-hand side in (4.10) can be further lower bounded by $1-\delta/2$ if M satisfies (4.8).

We next show that for large M, with high probability, the largest q terms of e_j, i.e. $e^{(1)}, \cdots, e^{(q)}$, coincide with $\{x_j\}_{j\in[q]}$. Particularly, denoting the index of $e^{(j)}$ as ℓ_j, we will show $[q] = \{\ell_1, \cdots, \ell_q\}$.

Note that $[q] = \{\ell_1, \cdots, \ell_q\}$ if and only if the following probabilistic event occurs:

$$C_q := \left\{ \min_{i \le q} e_i > \max_{j > q} e_j \right\}.$$

Since for every $i \le q$ and $j > q$, $e_i - e_j$ is a sum of M i.i.d. random variables Z, where $Z = 1$ with probability $\kappa(x_i)$, $Z = -1$ with probability $\kappa(x_j)$ and $Z = 0$ otherwise. Thus, we can bound the probability of C_q from below with another application of Hoeffding's inequality

$$\begin{aligned}
\mathbb{P}\left[C_q\right] = 1 - \mathbb{P}\left[C_q^{\complement}\right] &\ge 1 - \sum_{i \le q, j > q} \mathbb{P}\left[e_i - e_j < 0\right] \\
&\ge 1 - \sum_{i \le q, j > q} \mathbb{P}\left[e_i - e_j - \mathbb{E}[e_i - e_j] < -M\Delta\right] \\
&\ge 1 - \frac{h^2}{4} \exp\left(-\frac{M\Delta^2}{2}\right) \\
&\ge 1 - \frac{N^2}{4} \exp\left(-\frac{M\Delta^2}{2}\right),
\end{aligned}$$

which is lower bounded by $\delta/2$ if M satisfies (4.8). Taking a union bound, for M satisfying (4.8), both the event in (4.10) and C_q occur with probability at least $1 - \delta$.

To finish the proof, it suffices to show that conditional on both events, (i) $\sum_{i \in T} \kappa(x_i) \ge 1 - \eta/2$ and (ii) $|T| \le \max\{q, d+1\}$. Let $T_- = T \setminus \{\ell_L\}$. Conditional on the event in (4.10), it follows from the stopping rule in Algorithm 3 that

$$\begin{aligned}
\sum_{j \in T} \kappa(x_j) &\overset{(4.10)}{\ge} \sum_{j \in T \cap [q]} \left(\frac{1}{M} e_j - \frac{\eta}{18q}\right) + \sum_{j \in T \cap [N] \setminus [q]} \kappa(x_j) \\
&\ge \sum_{j \in T \cap [q]} \left(\frac{1}{M} e_j - \frac{\eta}{18q}\right) + \sum_{j \in [N] \setminus [q]} \kappa(x_j) - \sum_{j \in [N] \setminus [q]} \kappa(x_j) \\
&\overset{(4.10),(4.7)}{\ge} \sum_{j \in T \cap [q]} \left(\frac{1}{M} e_j - \frac{\eta}{18q}\right) + \sum_{j \in [N] \setminus [q]} \frac{1}{M} e_j - \frac{\eta}{18q} - \frac{\eta}{18} \\
&\ge \sum_{j \in T \cap [q]} \left(\frac{1}{M} e_j - \frac{\eta}{18q}\right) + \sum_{j \in T \cap [N] \setminus [q]} \frac{1}{M} e_j - \frac{\eta}{18q} - \frac{\eta}{18} \\
&\ge 1 - \frac{\eta}{3} - \frac{\eta}{18} - \frac{\eta}{18q} - \frac{\eta}{18} > 1 - \frac{\eta}{2},
\end{aligned}$$

which shows that (i) holds true.

Algorithm 4: Approximate Archetypal Analysis (AAA)

Input: $\{x_i\}_{i\in[N]}$: dataset, k: number of archetypes, p: approximation rank, s: Krylov subspace parameter, M: number of projections, η: approximation accuracy

Output: an approximate solution to (1.1)

1: generate p (Gaussian) random initializations: $\boldsymbol{S}\in\mathbb{R}^{N\times p}$, $\boldsymbol{S}_{ij}\overset{\text{i.i.d}}{\sim}\mathcal{N}(0,1)$
2: construct the Krylov subspace: $\boldsymbol{K}=[\boldsymbol{XS},(\boldsymbol{XX}^T)\boldsymbol{XS},\cdots,(\boldsymbol{XX}^T)^{s-1}\boldsymbol{XS}]\in\mathbb{R}^{d\times(sp)}$
3: compute the QR decomposition for $\boldsymbol{K}$: $\boldsymbol{K}=\boldsymbol{QR}$
4: compute the SVD of $\boldsymbol{X}_{\text{emd}}=\boldsymbol{X}^T\boldsymbol{Q}$: $\boldsymbol{X}_{\text{emd}}=\boldsymbol{U}_{\text{emd}}\Sigma_{\text{emd}}\boldsymbol{V}_{\text{emd}}^T$
5: form approximate reduced SVD representation: $\widetilde{\boldsymbol{X}}=\Sigma_{\text{emd}}[1:p,1:p](\boldsymbol{U}_{\text{emd}}[:,1:p])^T$
6: apply Algorithm 3 to $\widetilde{\boldsymbol{X}}$ with parameters (M,η) to find a subset of $T\subset[N]$
7: solve the reduced archetypal analysis problem:

$$(\widetilde{\boldsymbol{A}}_\star,\widetilde{\boldsymbol{B}}_\star)\in\arg\min_{\widetilde{\boldsymbol{A}}\in\mathbb{R}_{\text{cs}}^{|T|\times k},\widetilde{\boldsymbol{B}}\in\mathbb{R}_{\text{cs}}^{k\times N}}\frac{1}{\sqrt{N}}\|\widetilde{\boldsymbol{X}}-\widetilde{\boldsymbol{X}}_T\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F$$

8: extend $\widetilde{\boldsymbol{A}}_\star$ to an $\mathbb{R}^{N\times k}$ matrix by first creating a zero matrix $\boldsymbol{A}_{null}\in\mathbb{R}^{N\times k}$, then $\boldsymbol{A}_{null}[T,:]\leftarrow\widetilde{\boldsymbol{A}}_\star$, and finally $\widetilde{\boldsymbol{A}}_\star\leftarrow\boldsymbol{A}_{null}$
9: return $\boldsymbol{A}_\star$, $\boldsymbol{B}_\star$

To show (ii), it suffices to consider the case where $\min\left\{\ell:\frac{1}{M}\sum_{j\in[\ell]}e^{(j)}>1-\eta/3\right\}>d+1$, since otherwise $L=d+1\le\max\{q,d+1\}$. In this case, the stopping rule in Algorithm 3 tells us

$$\begin{aligned}\sum_{j\in T_-}\kappa(x_j)&\overset{(4.10)}{\le}\sum_{j\in T_-\cap[q]}\left(\frac{1}{M}e_j+\frac{\eta}{18q}\right)+\sum_{j\in T_-\cap[N]\setminus[q]}\kappa(x_j)\\&\overset{(4.7)}{\le}\sum_{j\in T_-\cap[q]}\left(\frac{1}{M}e_j+\frac{\eta}{18q}\right)+\frac{\eta}{18}\\&\le\sum_{j\in T_-}\frac{1}{M}e_j+\frac{\eta}{18}+\frac{\eta}{18}\overset{(4.7)}{<}\sum_{j\in[q]}\kappa(x_j).\end{aligned}\tag{4.11}$$

Further conditioning on C_q, we have $T_-\subset[q]$ or $[q]\subset T_-$. But the latter cannot happen owing to (4.11). This implies $|T|=|T_-|+1\le q=\max\{q,d+1\}$, establishing (ii). □

5. An approximate AA algorithm

Putting results in Section 3 and 4 together, we have the following approximate algorithm for archetypal analysis (AAA):

Under appropriate assumptions on the input parameters, we have the following guarantee for the solutions computed by Algorithm 4.

THEOREM 5.1. Under the same assumptions in Theorem 4.4 and $p \gtrsim \log(1/\delta)$, if

$$s \geq C \log\left(\frac{N}{\delta}\right) \qquad \eta = \left(\frac{\text{opt}(\boldsymbol{X})\varepsilon}{\sqrt{2}\pi \max_{i\in[N]} \|x_i\|_2}\right)^{p-1} \tag{5.1}$$

$$M \geq \max\left\{\frac{324q^2}{\eta^2}, \frac{4}{\Delta^2}\right\} \log\left(\frac{3N}{\sqrt{\delta}}\right), \tag{5.2}$$

where C is the same constant as in Theorem 3.6, $\text{opt}(\boldsymbol{X})$ is the optimum value of (1.1), q, Δ are the same as defined in Theorem 4.4, then with probability at least $1 - 2\delta$, $|T| \leq \max\{q, p+1\}$, and the approximate archetypes $\boldsymbol{X}\widetilde{\boldsymbol{A}}_\star$ as well as the coefficient matrix $\widetilde{\boldsymbol{B}}_\star$ returned by Algorithm 4 satisfy

$$\frac{1}{\sqrt{N}}\|\boldsymbol{X} - \boldsymbol{X}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F \leq (1+\varepsilon)\left(\text{opt}(\boldsymbol{X}) + 8\sigma_{p+1}(\boldsymbol{X})\right).$$

REMARK 5.1. According to Theorems 3.5 and 4.3, the computational complexity of data dimensionality reduction (step 1–5) and representation cardinality reduction (step 6) is $\mathcal{O}(dNp\log N + dp^2\log^2 N + Np\log N \min\{N, p\log N\})$ and $\mathcal{O}(MNp + N\log N) = \mathcal{O}\left((q^2\varepsilon^{-2(p-1)} + \Delta^{-2} + N)\log N\right)$, respectively. With probability at least $1 - 2\delta$, step 6 solves the reduced problem which has data dimension p and representation cardinality $|T| \leq \max\{p+1, q\}$. Thus, the overall complexity for Algorithm 4 is small if both p and q are small and Δ is away from 0. This corresponds to the scenario where $\boldsymbol{X}$ is approximately low-rank and has most of the curvature concentrated on a small subset.

Proof of Theorem 5.1 Let $(\boldsymbol{A}_\star, \boldsymbol{B}_\star)$ and $(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}})$ be solutions to (1.1) and

$$\min_{\boldsymbol{A}\in\mathbb{R}_{\text{cs}}^{N\times k}, \boldsymbol{B}\in\mathbb{R}_{\text{cs}}^{k\times N}} \frac{1}{\sqrt{N}}\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\boldsymbol{A}\boldsymbol{B}\|_F, \tag{5.3}$$

respectively. Under the assumptions on η and M, Theorems 4.2 and 4.4 together imply that with probability at least $1-\delta$, $|T| \leq \max\{p+1, q\}$ and

$$\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F \leq (1+\varepsilon)\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{B}}\|_F \leq (1+\varepsilon)\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{X}}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F. \tag{5.4}$$

Let $\widetilde{\boldsymbol{X}}_p = \widetilde{\boldsymbol{U}}_p\widetilde{\boldsymbol{X}}$ and $\widetilde{\boldsymbol{X}}_p = \boldsymbol{X} - \widetilde{\boldsymbol{X}}_p$, where $\widetilde{\boldsymbol{U}}_p$ is the left singular vector matrix of the low-rank approximation given by Algorithm 2. For s satisfying (5.1), it follows from Lemma 3.1 that with probability at least $1-\delta$,

$$\|\widetilde{\boldsymbol{X}}_{-p}\|_2 = \|\boldsymbol{X} - \widetilde{\boldsymbol{X}}_p\|_2 \leq 2\|\boldsymbol{X} - \boldsymbol{X}_p\|_2 = 2\sigma_{p+1}(\boldsymbol{X}), \tag{5.5}$$

where $\boldsymbol{X}_p$ is the best rank-p approximation for $\boldsymbol{X}$. Thus, both (5.4) and (5.5) hold with probability $1-2\delta$. Conditioning on (5.4) and (5.5), the rest of the proof is similar to the computation in (3.4)

$$\begin{aligned}
\|\boldsymbol{X}-\boldsymbol{X}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F &\le \|\widetilde{\boldsymbol{X}}_p-\widetilde{\boldsymbol{X}}_p\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F+\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F\\
&= \|\widetilde{\boldsymbol{X}}-\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F+\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F\\
&\overset{(5.4)}{\le} (1+\varepsilon)\|\widetilde{\boldsymbol{X}}-\widetilde{\boldsymbol{X}}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F\\
&= (1+\varepsilon)\|\widetilde{\boldsymbol{X}}_p-\widetilde{\boldsymbol{X}}_p\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F\\
&\le (1+\varepsilon)\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+(1+\varepsilon)\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+\|\widetilde{\boldsymbol{X}}_{-p}-\widetilde{\boldsymbol{X}}_{-p}\widetilde{\boldsymbol{A}}_\star\widetilde{\boldsymbol{B}}_\star\|_F\\
&\overset{(3.5)}{\le} (1+\varepsilon)\left(\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+2\|\widetilde{\boldsymbol{X}}_{-p}\|_F+2\sqrt{N}\|\widetilde{\boldsymbol{X}}_{-p}\|_2\right)\\
&\le (1+\varepsilon)\left(\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+4\sqrt{N}\|\widetilde{\boldsymbol{X}}_{-p}\|_2\right)\\
&\overset{(5.5)}{\le} (1+\varepsilon)\left(\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{A}_\star\boldsymbol{B}_\star\|_F+8\sigma_{p+1}(\boldsymbol{X})\sqrt{N}\right).
\end{aligned}$$

Dividing both sides by $\sqrt{N}$ yields the desired result. □

6. Numerical experiments

In this section, we apply the proposed algorithm (Algorithm 4) to compute the archetypes for three real datasets, including a time series dataset and two image datasets. When implementing the alternating minimization algorithm for solving AA, we use the k-means to find an initial guess for the archetypes; the subproblems are solved using the existing package '`quadprog`' [42] in `R` [35]. The algorithm stops if the relative objective decrease falls below `1e-3`. We will compare the computation time and accuracy of the following algorithms:

- SVD-AA: Alternating minimization applied to the reduced singular value representation of $\boldsymbol{X}$ as in (3.1), where truncation keeps 99.99% of the variance of the data. SVD is implemented using the built-in function '`svd`' in `R`.
- AAA: Approximate archetypal analysis (Algorithm 4), with $s = \lceil \log N \rceil$.
- archetypes: A function for AA in the package `archetypes` in `R` [15, 16], whose implementation is different from Algorithm 1.

To ensure comparability of the results, we do not include other accelerated algorithms such as the active-subset solver [6] and the coreset approximation [28], which have a different focus as opposed to our methods. All reported results in this section were obtained on a Macbook Air with an M1 processor and 8GB of RAM.

6.1 *S&P 500 cumulative log-returns*

The Standard and Poor's 500 (S&P 500) is a stock market index consisting of 500 large companies listed on stock exchanges in the USA. It is one of the most commonly used equity indices to evaluate

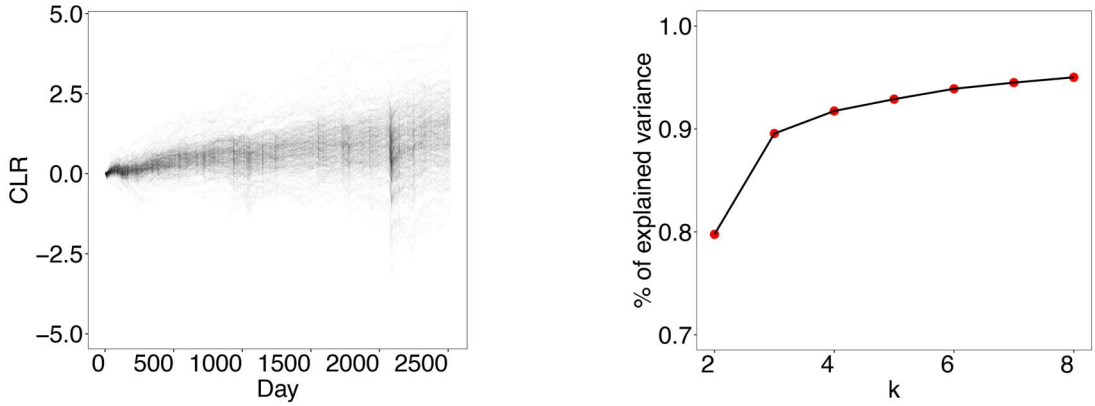

Figure 2. The CLR of 385 S&P 500 stocks from December 2011 to December 2021 (left). Variances of $\boldsymbol{X}$ explained by the k archetypes identified by AA as a function of k for $k = 2, \cdots, 8$ (right).

the financial market as well as the economy. The companies that are selected for the S&P 500 index are changing with time. In this example, we consider a dataset comprised of 385 companies that are in the S&P 500 index by January 2022, with close prices recorded from December 2011 to December 2021. We compute for each column in $\boldsymbol{X}$ a 2515-dimensional time series representing the cumulative log-return (CLR) of a company over 10 years. The CLR is calculated on a daily basis using the adjusted prices of stocks. Visualization of the dataset is given in the first plot in Fig. 2. In the rest of the section, we assume that the CLR of each company in the S&P 500 index can be decomposed with respect to a few distinct growth patterns that can be identified via AA.

To apply AA, we need to first determine the number of archetypes k. Like other unsupervised learning methods, there is no principled rule to find the correct number of k for real-life datasets. A more practical solution is to follow the heuristic 'elbow rule' [39] to choose k approximately. In this case, we apply SVD-AA to find such a k. In particular, we plot the variance of the dataset explained by the archetypes given by SVD-AA as a function of k (see Fig. 2) and choose k to be the point where the curve starts to plateau, which is around $k = 3$.

Setting $k = 3$, we apply SVD-AA, AAA and archetypes to compute the archetypes for $\boldsymbol{X}$. The parameters p, M and η in AAA are set as 20, 10 000 and 0.003 (so that $\eta/3 = 0.001$), respectively. Each experiment is repeated 100 times, with the learned archetypes (in the first 10 experiments), the running times (elapsed time computed using the '`system.time()`' function in `R`) and residuals reported in Figs 3 and 4, respectively.

It can be seen from Fig. 4 that SVD-AA, as expected, gives the best-computed archetypes in terms of the residual on average; however, its computation time is significantly longer than the other two methods. The built-in function archetypes has the worst performance, and its computation time is between the other two methods. The AAA, which first reduces the dimension of the dataset before applying the alternating minimization, achieves competitive results with SVD-AA but takes much less time (more than 30 times faster than SVD-AA). This may be because $\boldsymbol{X}$ is essentially low-dimensional and admits a parsimonious approximation for its convex hull. A numerical justification for this argument can be seen from the spectral decay of the sample covariance matrix of $\boldsymbol{X}$ as well as the scatterplot of the reduced representation of $\boldsymbol{X}$ with respect to the first two principal components (PCs), as illustrated in Fig. 5.

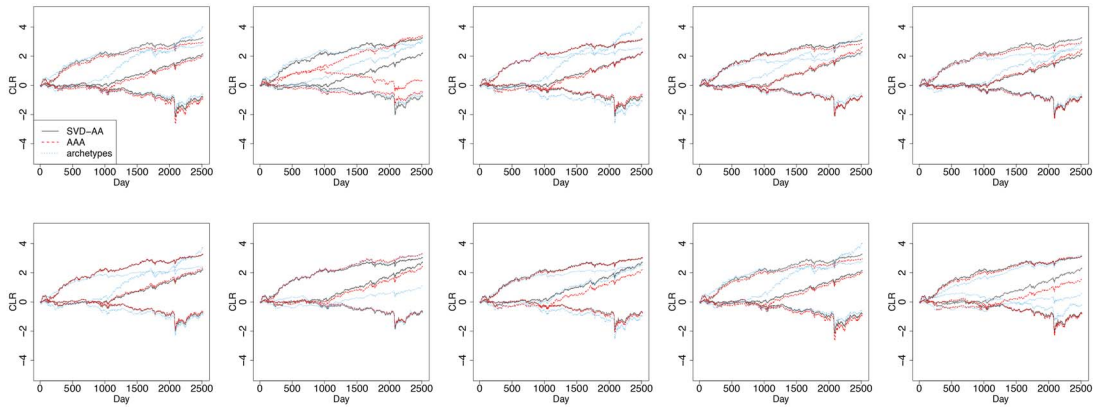

Figure 3. Instances of the computed archetypes by SVD-AA, AAA and archetypes in the first 10 experiments.

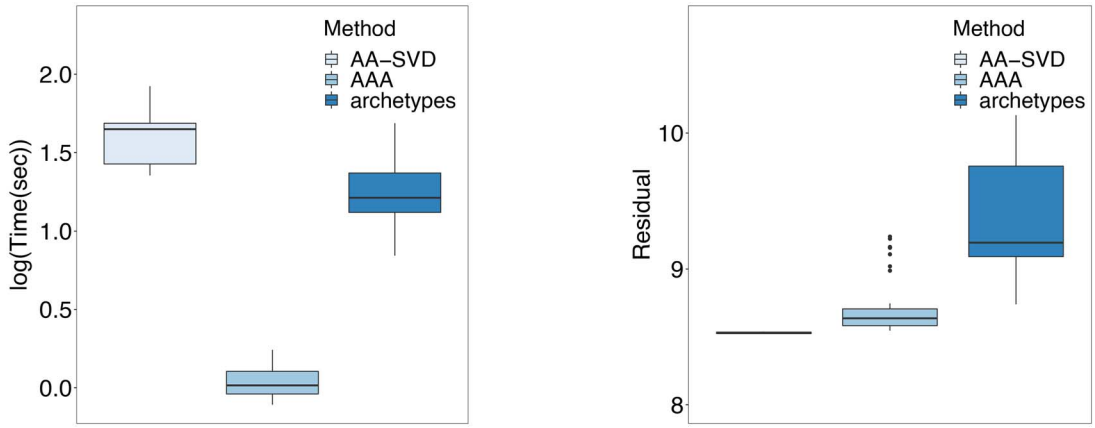

Figure 4. Boxplots of the running times (left) and residuals (right) of SVD-AA, AAA and archetypes in 100 experiments.

To implement AAA, it is necessary to choose the input parameters in advance. The optimal choice for the parameters is problem-dependent and often there is no universal tuning strategy for it. The Krylov subspace parameter s is set as $\lceil \log N \rceil$ deterministically. We investigate the accuracy/running time dependence on p, M and η. In particular, we will use the same parameters as in the previous simulation. Whenever we test the dependence on one parameter, the other two are set fixed. We will test p, M and η at three different values, respectively, i.e. $p = 10, 20, 30$, $M = 10^3, 10^4, 10^5$ and $\eta = 0.3, 0.03, 0.003$. The results are given in Fig. 6.

Figure 6 shows that for the S&P 500 dataset, the accuracy of AAA has a strong dependence on η, which measures the missing proportion of curvature in the approximate convex hull construction. The number of random projections M mostly influences the running time while having only a mild impact on the accuracy when $p = 20$. The approximation rank p, as long as set reasonably large, is sufficient to give a good approximation result.

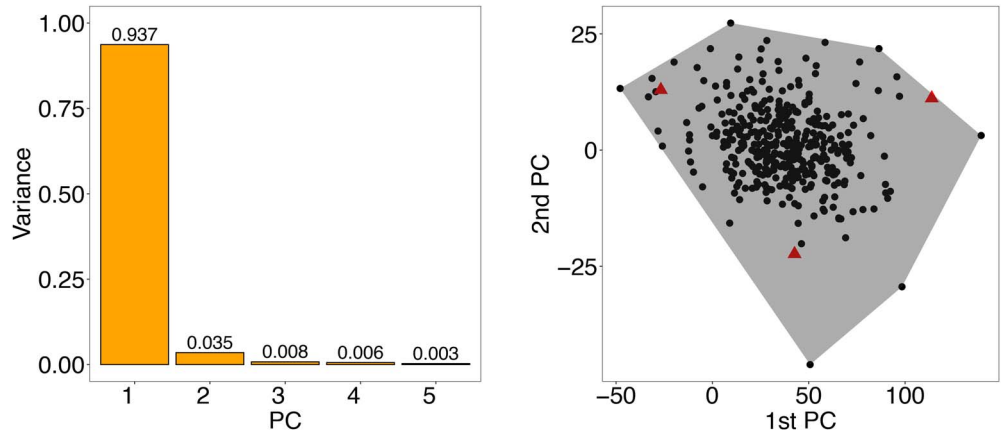

Figure 5. Variances explained by the first five principal components of $\boldsymbol{X}$ (left). Scatterplot of the reduced representation of $\boldsymbol{X}$ with respect to the first two principal components (which account for 97% of the variation of the dataset) and its convex hull. The red triangles are the reduced representation of the three archetypes (right).

In this example, we compare the three archetypes with the same number of centers identified by the k-means; see the first plot in Fig. 7. It can be seen that the archetypal curves are visually more illustrative than the centers of the k-means, which share a similar growth pattern with differing slopes. Indeed, the percentage of variance explained by AA is around 90%; the same number for the k-means and PCA are 64% and 98%, respectively. Visualization of the convex coefficients for each data point with respect to the three archetypes is given in the ternary plot in Fig. 7. In this case, most of the data fall in the interior of the simplex, suggesting that the S&P dataset can be well summarized using a polytopic structure.

To further understand the meanings of the three archetypes, for each archetype, we single out the tickers of the top five companies having the largest component in the following direction

- A1: NFLX: $(\mathbf{1}, 0, 0)$, STZ.B: $(\mathbf{0.9}, 0.1, 0)$, ILMN: $(\mathbf{0.89}, 0.03, 0.08)$, REGN: $(\mathbf{0.84}, 0.16, 0)$, FLT: $(\mathbf{0.83}, 0.17, 0)$;
- A2: AMD: $(0, \mathbf{1}, 0)$, FTNT: $(0, \mathbf{0.89}, 0.11)$, ISRG: $(0.02, \mathbf{0.83}, 0.15)$, LRCX: $(0.17, \mathbf{0.83}, 0)$, CPRT: $(0.18, \mathbf{0.81}, 0)$;
- A3: MRO: $(0, 0, \mathbf{1})$, DVN: $(0, 0, \mathbf{1})$, OXY: $(0, 0, \mathbf{1})$, APA: $(0, 0, \mathbf{1})$, MOS: $(0, 0.03, \mathbf{0.97})$.

All the five companies in A3 are in the energy industry (oil, mining, etc.), representing the traditional aspect of the financial market. Companies in A1 and A2 leave more room for interpretation. In particular, for A1, NFLX is an entertainment company, STZ.B is a food company (beer and wine), ILMN is a biotechnology company, REGN is a pharmaceutical company and FLT is a financial service company; for A2, both AMD and LRCX are in the semiconductor industry, FTNT is a cybersecurity company, ISRG is a surgical equipment design company and CPRT is a company that provides online vehicle auction and remarketing services. According to the quant ratings on `https://seekingalpha.com/` between 2021 and 2022, all these companies have high profitability; each of these companies has consecutively ranked above A- and many have been A+ in the three latest reports (the factor grade ranges from A+ to F). This feature is also manifested in the upward trend in the archetypal curves associated with A1 and A2. Moreover, they are more resilient than the traditional industries when

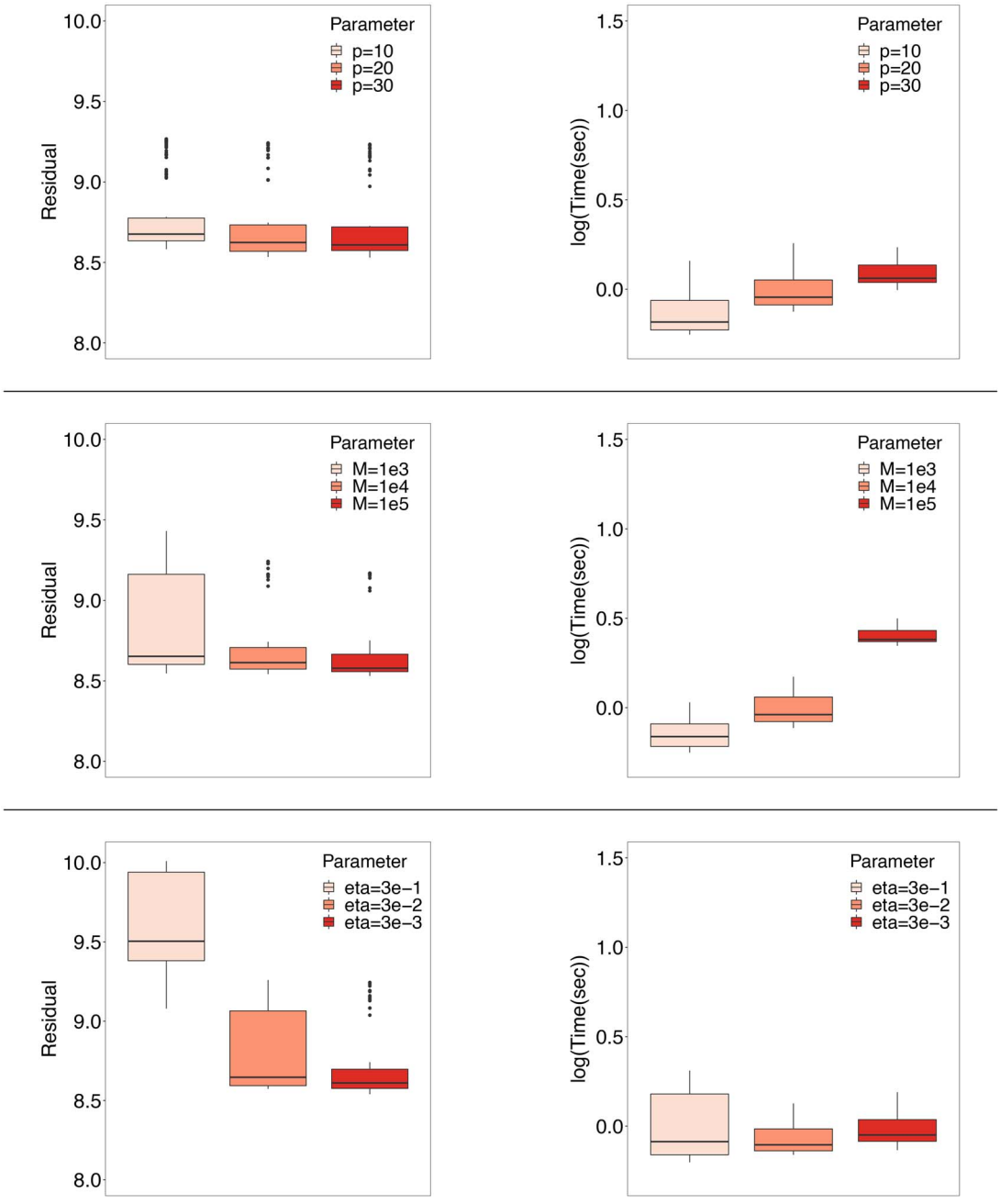

Figure 6. Accuracy/running time dependence of AAA algorithm for the S&P 500 dataset with base parameters $p = 20, M = 10^4$ and $\eta = 0.003$.

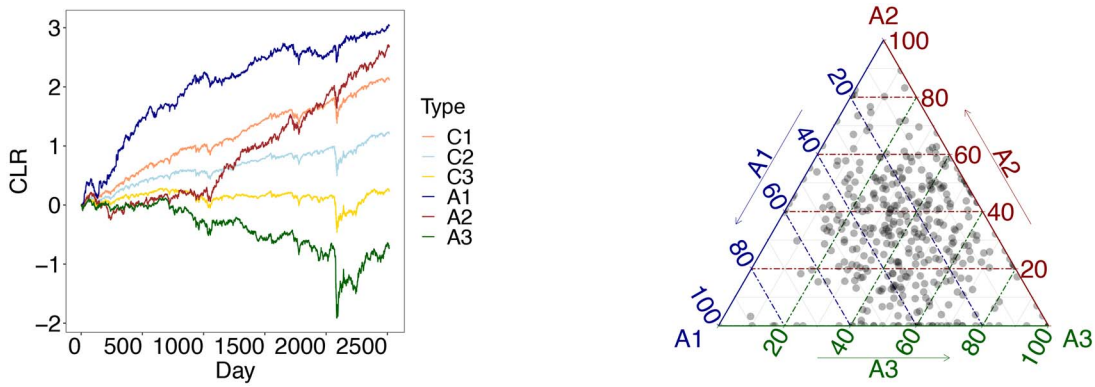

Figure 7. Comparison of the three centers (C1, C2, C3) given by the k-means and the three archetypes (A1, A2, A3) given by the SVD-AA for the S&P 500 dataset (left). Visualization of the convex combination coefficients of each data point with respect to the three archetypes (right).

Figure 8. Six images in the dataset with largest component in each archetypal direction identified by AAA (top) compared with the six images in the dataset closest to the centers of the k-means (bottom).

unexpected events occur (e.g. Covid-19 pandemic in early 2020), as can be seen from the 'V' shape of these curves near Day 2000 in the first plot in Fig. 7. The difference between A1 and A2 is more difficult to corroborate using recent financial data. From a macroscopic perspective, A1 represents the more established highly profitable industries in the market; they maintain a steady pace of CLR growth over time. A2 represents the emerging industries that, while not as profitable as A1, possess relatively more growth potential. This conclusion can be numerically inspected by comparing the slope of the A1 and A2 curves in Fig. 7.

6.2 *Intel image*

The Intel Image dataset [21] has been used for multi-class classification in machine learning, and consists of 24 000 images representing 6 different categories of the scene: 'Buildings', 'Forest', 'Glacier', 'Mountain', 'Sea' and 'Street'. Each image is a 150 × 150 pixel color image, which corresponds to a 67 500-dimensional vector through vectorization and stacking of the pixel matrices ($d = 67\,500$). We randomly select 3000 samples in the training dataset ($N = 3000$) and apply AAA

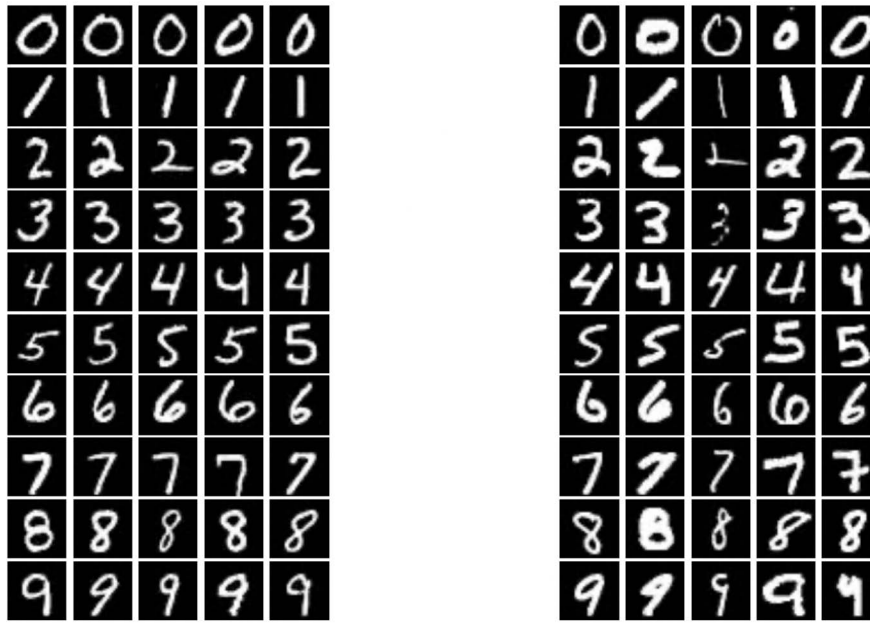

Figure 9. Typical patterns (left) and archetypes (right) idenfied by the k-means and AAA in each label class in the MNIST training dataset.

to extract representative patterns. Note we could have used the full dataset; however, this would require using a more efficient optimization solver for the subproblems to ensure the computation is done in a reasonable time. Since there are 6 different categories of images, we set $k = 6$. The input parameters for AAA are chosen as $p = 30$, $M = 10^4$ and $\eta = 0.03$. We compare the computed archetypes given by AAA with the clustering centers given by the k-means in Fig. 8.

In this experiment, the instance running time is 527.356s (107.147s for data dimensionality reduction, 1.627s for representation cardinality reduction and 418.582s for solving the reduced problem using Algorithm 1) for AAA, and 151.787s for the k-means. The running times are random due to the random nature of the algorithms. In this case, the cardinality of the extreme points used to build up the approximate convex hull is 738. The 6 archetypes account for about 41.2% of the variance of the dataset, as opposed to 28.3% explained by the k-means. The other two methods, SVD-AA and archetypes, cannot be implemented within a reasonable time.

For each archetype, we find the image that has the largest component with respect to it in the dataset. We also identify the images closest to the k-means centers. The results are reported in Fig. 8. According to the label information, the images on the top and bottom panels in Fig. 8 (from left to right) correspond to 'Forest', 'Buildings', 'Glacier', 'Glacier', 'Street', 'Sea' and 'Mountain', 'Mountain', 'Mountain', 'Sea', 'Glacier' and 'Forest', respectively. Despite an approximate algorithm, AAA produces more diversified results than the k-means in terms of image content. The only repetition occurs in the third and fourth pictures, where both the snow mountains are classified as 'Glacier'.

6.3 *MNIST dataset*

The MNIST database [26] is a large database of handwritten digits that is commonly used for both classification and clustering tasks. Each data point in MNIST is a 28×28 gray-scale image (i.e. a

784-dimensional vector) representing handwritten digits from 0 to 9. The total size of the training dataset is 42 000.

In this experiment, we use both the k-means and AA to analyze the data structure in each label class separately. We first split the training data into 10 different datasets corresponding to labeled digits $0, \cdots, 9$, respectively, each having a size of around 4000. We apply both the k-means and AAA to the split datasets to identify the typical patterns and the archetypes, respectively. After running the 'elbow inspection' for the k-means at different labels, we found $k = 5$ to be a reasonable choice on average. To be consistent, we also use $k = 5$ for AAA. Moreover, the other parameters in AAA are set as $p = 10$, $M = 10^4$ and $\eta = 0.03$. As before, in each label class, we find the images in the corresponding datasets that are closest to the k-means centers as well as have the largest convex combination coefficients with respect to the archetypes. The results are reported in Fig. 9.

In general, the k-means centers are the images that are representative of each label class. They are more standard and usually can be distinguished by raws eyes. The approximate archetypes found by AAA are more extreme in terms of size, shape, position, etc.

Data availability statement

The data underlying this article are available on the referenced online sources and in the supplementary material.

Acknowledgements

We would like to thank the anonymous referees for their very helpful comments which significantly improved the presentation of the paper. We would like to thank Yu Zhu for providing us with the S&P 500 dataset and helping clarify some related questions. We also thank Akil Narayan for reading through an early version of the draft and for providing several constructive comments. Y. Xu would like to thank the organizers of the MSRI Summer Graduate School on Mathematics of Big Data: Sketching and (Multi-) Linear Algebra for motivating discussions.

Funding

Hong Kong Research Grants Council (14301821 to R.H.); Direct Grants for Research, The Chinese University of Hong Kong; National Science Foundation (DMS-1752202 to B.O.); National Natural Science Foundation of China (12101524 to D.W.); University Development Fund from The Chinese University of Hong Kong, Shenzhen (UDF01001803 to D.W.); National Science Foundation (DMS-1848508 to Y.X.).

Reference

1. Abrol, V. & Sharma, P. (2020) A geometric approach to archetypal analysis via sparse projections. in *International Conference on Machine Learning*, pp. 42–51. PMLR.
2. Avron, H., Maymounkov, P. & Toledo, S. (2010) Blendenpik: supercharging LAPACK's least-squares solver. *SIAM J. Sci. Comput.*, **32**, 1217–1236.
3. Battaglino, C., Ballard, G. & Kolda, T. G. (2018) A practical randomized CP tensor decomposition. *SIAM J. Matrix Anal. Appl.*, **39**, 876–901.
4. Bauckhage, C., Kersting, K., Hoppe, F. & Thurau, C. (2015) Archetypal analysis as an autoencoder. *Workshop New Challenges in Neural Computation*. Citeseer, p. 8.

5. Boutsidis, C., Zouzias, A. & Drineas, P. (2010) Random projections for *k*-means clustering. *Advances in Neural Information Processing Systems*, **23**, 298–306.
6. Chen, Y., Mairal, J. & Harchaoui, Z. (2014) Fast and robust archetypal analysis for representation learning. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1478–1485.
7. Clarkson, K. L. & Woodruff, D. P. (2017) Low-rank approximation and regression in input sparsity time. *J. ACM*, **63**, 1–45.
8. Cohen, M. B., Elder, S., Musco, C., Musco, C. & Persu, M. (2015) Dimensionality reduction for k-means clustering and low rank approximation. in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 163–172.
9. Cohen, M. B., Musco, C. & Musco, C. (2017) Input sparsity time low-rank approximation via ridge leverage score sampling. in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1758–1777. SIAM.
10. Cutler, A. & Breiman, L. (1994) Archetypal analysis. *Dent. Tech.*, **36**, 338–347.
11. Damle, A. & Sun, Y. (2017) A geometric approach to archetypal analysis and nonnegative matrix factorization. *Dent. Tech.*, **59**, 361–370.
12. Drineas, P., Mahoney, M. W., Muthukrishnan, S. & Sarlós, T. (2011) Faster least squares approximation. *Numer. Math.*, **117**, 219–249.
13. Eckart, C. & Young, G. (1936) The approximation of one matrix by another of lower rank. *Psychometrika*, **1**, 211–218.
14. Erichson, N. B., Mendible, A., Wihlborn, S. & Kutz, J. N. (2018) Randomized nonnegative matrix factorization. *Pattern Recognition Letters*, **104**, 1–7.
15. Eugster, M. J. A. & Leisch, F. (2009) From spider-man to hero – archetypal Analsis in R. *J. Stat. Softw.*, **30**, 1–23.
16. Eugster, M. J. A. & Leisch, F. (2011) Weighted and robust archetypal analysis. *Comput. Statist. Data Anal.*, **55**, 1215–1225.
17. Graham, R. & Oberman, A. M. (2017) Approximate convex hulls: sketching the convex hull using curvature. *arXiv preprint arXiv:1703.01350*.
18. Halko, N., Martinsson, P.-G. & Tropp, J. A. (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, **53**, 217–288.
19. Hastie, T., Tibshirani, R. & Friedman, J. (2001) *The Elements of Statistical Learning, Springer Series in Statistics*. New York, NY, USA: Springer New York Inc.
20. Hoeffding, W. (1963) Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, **58**, 13–30.
21. Intel (2019) *Intel Image Classification Challenge*. https://www.kaggle.com/puneet6060/intel-image-classification.
22. Javadi, H. & Montanari, A. (2020) Nonnegative matrix factorization via archetypal analysis. *J. Amer. Statist. Assoc.*, **115**, 896–907.
23. Knuth, D. E. (1997) *The art of computer programming*, vol. **3**. Pearson Education.
24. Kozlov, M. K., Tarasov, S. P. & Khachiyan, L. G. (1979) Polynomial solvability of convex quadratic programming. *Doklady Akademii Nauk*, vol. **248**. Russian Academy of Sciences, pp. 1049–1051.
25. Lax, P. D. (2002) *Functional Analysis*. John Wiley & Sons. Inc. Publication.
26. LeCun, Y. (1998) The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
27. Mair, S., Boubekki, A. & Brefeld, U. (2017) Frame-based data factorizations. in *International Conference on Machine Learning*, pp. 2305–2313. PMLR.
28. Mair, S. & Brefeld, U. (2019) Coresets for archetypal analysis. *Advances in Neural Information Processing Systems*, **32**, 7247–7255.
29. Makarychev, K., Makarychev, Y. & Razenshteyn, I. (2019) Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1027–1038.

30. Mei, J., Wang, C. & Zeng, W. (2018) Online dictionary learning for approximate archetypal analysis. in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 486–501.
31. Mørup, M. & Hansen, L. K. (2012) Archetypal analysis for machine learning and data mining. *Neurocomputing*, **80**, 54–63.
32. Musco, C. & Musco, C. (2015) Randomized block Krylov methods for stronger and faster approximate singular value decomposition. *Advances in Neural Information Processing Systems*, **2015**, 1396–1404.
33. Osting, B., Wang, D., Xu, Y. & Zosso, D. (2021) Consistency of archetypal analysis. *SIAM J. Math. Data Sci.*, **3**, 1–30.
34. Qian, Y., Tan, C., Mamoulis, N. & Cheung, D. W. (2018) Dsanls: Accelerating distributed nonnegative matrix factorization via sketching. in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 450–458.
35. R Core Team (2020) *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
36. Rudelson, M. & Vershynin, R. (2008) The Littlewood–Offord problem and invertibility of random matrices. *Adv. Math.*, **218**, 600–633.
37. Sarlos, T. (2006) Improved approximation algorithms for large matrices via random projections. in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 143–152. IEEE.
38. Shoval, O., Sheftel, H., Shinar, G., Hart, Y., Ramote, O., Mayo, A., Dekel, E., Kavanagh, K. & Alon, U. (2012) Evolutionary trade-offs, Pareto optimality, and the geometry of phenotype space. *Science*, **336**, 1157–1160.
39. Thorndike, R. L. (1953) *Psychometrika*, *Psychometrika*, pp. 267–276.
40. Thurau, C., Kersting, K., Wahabzada, M. & Bauckhage, C. (2011) Convex non-negative matrix factorization for massive datasets. *Knowledge and information systems*, **29**, 457–478.
41. Tropp, J. A., Yurtsever, A., Udell, M. & Cevher, V. (2017) Practical sketching algorithms for low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.*, **38**, 1454–1485.
42. Turlach, B. A. & Weingessel, A. (2011) Quadprog: functions to solve quadratic programming problems. *R package*, version 1.5–4.
43. Vershynin, R. (2018) *High-dimensional probability: An introduction with applications in data science*, vol. **47**. Cambridge university press.
44. Wang, Y., Tung, H.-Y., Smola, A. J. & Anandkumar, A. (2015) Fast and guaranteed tensor decomposition via sketching. *Advances in neural information processing systems*, **28**, 2512–2520.
45. Woodruff, D. P. (2014) Sketching as a tool for numerical linear algebra. *Foundations and trends® Theoret. Comput. Sci.*, **10**, 1–157.

A. Derivation of (2.2)

The Gauss-Seidel method updates the identified archetypes (i.e. the columns of $\boldsymbol{Z}$) one at a time. In the i-th step, the procedure optimizes over the i-th column of $\boldsymbol{Z}$ with the rest kept fixed. It can be verified from direct computation that for $i \in [k]$,

$$\begin{aligned}
\|\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{B}\|_F^2 &= \sum_{j\in[d]}\sum_{\ell\in[N]}\left[\boldsymbol{X}[j,\ell]^2 - 2\boldsymbol{X}[j,\ell]\sum_{s\in[k]}\boldsymbol{Z}[j,s]\boldsymbol{B}[s,\ell] + \left(\sum_{s\in[k]}\boldsymbol{Z}[j,s]\boldsymbol{B}[s,\ell]\right)^2\right] \\
&= \sum_{j\in[d]}\sum_{\ell\in[N]}\left[(\boldsymbol{Z}[j,i]\boldsymbol{B}[i,\ell])^2 - 2\boldsymbol{Z}[j,i]\boldsymbol{B}[i,\ell]\left(\boldsymbol{X}[j,\ell] - \sum_{s\neq i}\boldsymbol{Z}[j,s]\boldsymbol{B}[s,\ell]\right)\right] + \Delta \\
&= \|\boldsymbol{B}[i,:]\|_2^2\sum_{j\in[d]}\left[\boldsymbol{Z}[j,i] - \frac{1}{\|\boldsymbol{B}[i,:]\|_2^2}\sum_{\ell\in[N]}\boldsymbol{B}[i,\ell]\left(\boldsymbol{X}[j,\ell] - \sum_{s\neq i}\boldsymbol{Z}[j,s]\boldsymbol{B}[s,\ell]\right)\right]^2 + \Delta \\
&= \|\boldsymbol{B}[i,:]\|_2^2\left\|\boldsymbol{Z}[:,i] - \frac{\boldsymbol{D}_i(\boldsymbol{B}[i,:])^T}{\|\boldsymbol{B}[i,:]\|_2^2}\right\|_2^2 + \Delta,
\end{aligned}$$

where $\boldsymbol{D}_i = \boldsymbol{X} - \boldsymbol{Z}[:,-i]\boldsymbol{B}[-i,:]$ and Δ collects the terms that do not depend on $\boldsymbol{Z}[:,i]$. Since $\boldsymbol{Z}[:,i] = \boldsymbol{X}\boldsymbol{A}[:,i]$, minimizing $\|\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{B}\|_F^2$ is equivalent to solving

$$\min_{a\in\mathbb{R}^N,\|a\|_1=1,a\geq 0}\left\|\boldsymbol{X}a - \frac{\boldsymbol{D}_i(\boldsymbol{B}[i,:])^T}{\|\boldsymbol{B}[i,:]\|_2^2}\right\|_2^2 \qquad i\in[k]. \tag{A.1}$$