Journal of Machine Learning Research 19 (2018) 1-30 Submitted 2/18; Revised 9/18; Published 11/18

Clustering is semidefinitely not that hard: Nonnegative SDP for manifold disentangling

Mariano Tepper MTEPPER@FLATIRONINSTITUTE.ORG
Flatiron Institute, Simons Foundation, NY, USA

Anirvan M. Sengupta ANIRVANS@PHYSICS.RUTGERS.EDU
Dept. of Physics and Astronomy, Rutgers University, NJ, USA
Flatiron Institute, Simons Foundation, NY, USA

Dmitri Chklovskii DCHKLOVSKII@FLATIRONINSTITUTE.ORG
Flatiron Institute, Simons Foundation, NY, USA
NYU Langone Medical Center, New York, NY

Editor: Daniel Lee

Abstract

In solving hard computational problems, semidefinite program (SDP) relaxations often play an important role because they come with a guarantee of optimality. Here, we focus on a popular semidefinite relaxation of K-means clustering which yields the same solution as the non-convex original formulation for well segregated datasets. We report an unexpected finding: when data contains (greater than zero-dimensional) manifolds, the SDP solution captures such geometrical structures. Unlike traditional manifold embedding techniques, our approach does not rely on manually defining a kernel but rather enforces locality via a nonnegativity constraint. We thus call our approach NOnnegative MAnifold Disentangling, or NOMAD. To build an intuitive understanding of its manifold learning capabilities, we develop a theoretical analysis of NOMAD on idealized datasets. While NOMAD is convex and the globally optimal solution can be found by generic SDP solvers with polynomial time complexity, they are too slow for modern datasets. To address this problem, we analyze a non-convex heuristic and present a new, convex and yet efficient, algorithm, based on the conditional gradient method. Our results render NOMAD a versatile, understandable, and powerful tool for manifold learning.

Keywords: K-means, semidefinite programming, manifolds, conditional gradient method

1. Introduction

In the quest for an algorithmic theory of biological neural networks, some of the authors have recently proposed a soft K-means clustering network that may model insect olfactory processing and other computations (Pehlevan et al., 2017). This network was derived by performing online optimization on the non-convex K-means objective function. Whereas the network dynamics and learning rules are biologically plausible, the nonconvexity of the objective makes it difficult to analyze the solutions and algorithm convergence.

Here, to understand the solutions computed by the clustering neural network, we consider a convex SDP relaxation of K-means (Kulis et al., 2007; Peng and Wei, 2007; Awasthi et al., 2015). Given data points $\{\mathbf{x}_i\}_{i=1}^n$ we define the Gramian matrix, $\mathbf{D}$, such that $(\mathbf{D})_{ij} = \mathbf{x}_i{}^\top \mathbf{x}_j$. Then, we

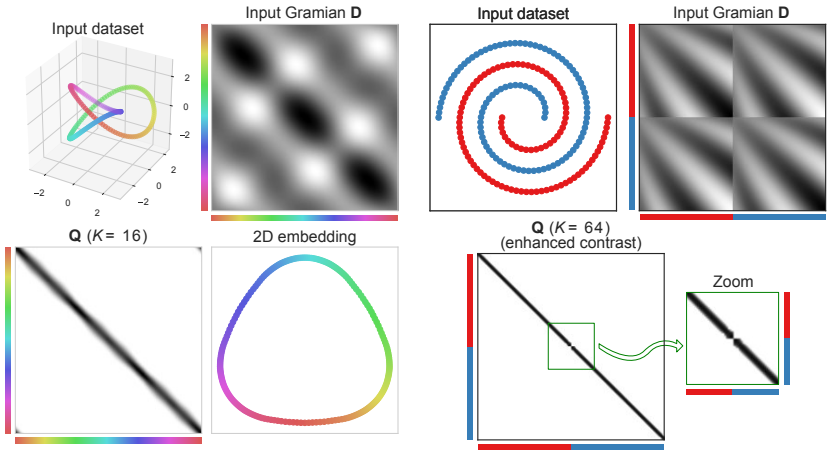


Figure 1: NOMAD, originally introduced as a convex relaxation of K-means clustering, surprisingly learns manifold structures in the data. (a) Learning the manifold of a trefoil knot, which cannot be "untied" in 3D without cutting it. NOMAD understands that this is a closed manifold, yielding a circulant matrix $\mathbf{Q}$, which can be "unfolded" in 2D. (b) Learning multiple manifolds with NOMAD. Although they are linearly non-separable, NOMAD correctly finds two submatrices, one for each manifold (for visual clarity, we enhance the contrast of $\mathbf{Q}$).

search for a cluster co-association matrix $\mathbf{Q}$, such that $(\mathbf{Q})_{ij} = 1$ if points i and j belong to the same cluster and $(\mathbf{Q})_{ij} = 0$ if they do not. The optimum $\mathbf{Q}_*$ can be found by solving the following optimization problem (the acronym will be explained below):

$$\mathbf{Q}_* = \underset{\mathbf{Q}\in\mathbb{R}^{n\times n}}{\operatorname{argmax}} \operatorname{Tr}(\mathbf{DQ}) \quad \text{s.t.} \quad \mathbf{Q1} = \mathbf{1},\ \operatorname{Tr}(\mathbf{Q}) = K,\ \mathbf{Q} \succeq 0,\ \mathbf{Q} \geq \mathbf{0}. \tag{NOMAD}$$

Its link with the original K-means clustering formulation is explained in Appendix A.[1]

First, we focus on the question: what does NOMAD compute? Until now, theoretical efforts have concentrated on showing that NOMAD is a good surrogate for K-means. Awasthi et al. (2015) study its solutions on datasets consisting of linearly separable clusters and demonstrate that they reproduce hard-clustering assignments of K-means. Moreover, the solution to NOMAD achieves hard clustering even for some datasets on which Lloyd's algorithm (Lloyd, 1982) fails (i.e., Iguchi et al. (2015); Mixon et al. (2016)). Related problems have been studied by Amini and Levina (2014); Javanmard et al. (2015); Yu et al. (2012).

In this work, we analyze NOMAD in a different regime than previous studies. Instead of focusing on cases and parameter settings where it approximates the original K-means formulation, we concentrate on alternative settings and discover that NOMAD is not merely a convex K-means imitator. NOMAD finds the manifold structure in the data, even discriminating different manifolds. Fig. 1 shows two examples of this unexpected behavior where NOMAD dissects the geometry of

1. **Notation.** $(\mathbf{X})_{ij}$, $(\mathbf{X})_{:j}$, $(\mathbf{X})_{i:}$ denote the (i,j)-th entry of matrix $\mathbf{X}$, the j-th column of $\mathbf{X}$, and the i-th row of $\mathbf{X}$, respectively. For vectors, we employ lowercase and we use a similar notation but with a single index. We write $\mathbf{X} \geq \mathbf{0}$ if a matrix $\mathbf{X}$ is entry-wise nonnegative and $\mathbf{X} \succeq \mathbf{0}$ if it is positive semidefinite.

the data. Because of this and of the central role played by the nonnegativity constraint in the SDP we call it a NOnnegative MAnifold Disentangling (NOMAD).

The next question is: how can we compute these solutions? Despite the theoretical advantages of convex optimization, in practice, the use of SDPs for clustering has remained limited. This is mainly due to the lack of efficient algorithms to solve the convex optimization problem. We address this issue by presenting an efficient convex solver for NOMAD, based on the conditional gradient method. The new algorithm can handle large datasets, extending the applicability of NOMAD to more interesting and challenging scenarios.

Organization. We first study the behavior of NOMAD theoretically by analyzing its solution for a simple synthetic example of a regular manifold with symmetry (Sec. 2). In this context, we demonstrate how NOMAD departs from standard K-means. Building on this analysis, we suggest that NOMAD has non-trivial manifold learning capabilities (Sec. 3) and demonstrate numerically NOMAD's good performance in non-trivial examples, including synthetic and real datasets. Then, motivated by the relatively slow performance of standard SDP solvers, we focus on scaling NOMAD to large modern datasets. In Sec. 4, we study both theoretically and experimentally an heuristic non-convex Burer-Monteiro-style algorithm (Kulis et al., 2007). Finally, we present a new convex and yet efficient algorithm for NOMAD. This algorithm allows us, for the first time, to study provable solutions of NOMAD on large datasets. Our software is publicly available at `https://github.com/simonsfoundation/sdp_kmeans`.

2. Theoretical analysis of manifold learning capabilities of NOMAD

Starting with the appearance of Isomap (Tenenbaum et al., 2000) and locally-linear embedding (LLE) (Roweis and Saul, 2000), there has been outstanding progress in the area of manifold learning (e.g., Belkin and Niyogi, 2003; Hadsell et al., 2006; Weinberger and Saul, 2006; Weiss et al., 2008). For a data matrix $\mathbf{X} = [\mathbf{x}_i]_{i=1}^n$ of column-vectors/points $\mathbf{x}_i \in \mathbb{R}^d$, the majority of these modern methods have three steps:

1. Determine the neighbors of each point. This can be done in two ways: (1) keep all point within some fixed radius ρ or (2) compute κ nearest neighbors.
2. Construct a weighting matrix $\mathbf{W}$, where $(\mathbf{W})_{ij} = 0$ if points i and j are not neighbors, and $(\mathbf{W})_{ij}$ is inversely proportional to the distance between points i and j otherwise.
3. Compute an embedding from $\mathbf{W}$ that is locally isometric to $\mathbf{X}$.

For the third step, many different and powerful approaches have been proposed, from computing shortest paths on a graph (Tenenbaum et al., 2000), to using graph spectral methods (Belkin and Niyogi, 2003), to using neural networks (Hadsell et al., 2006).

However, the success of these techniques depends critically on the ability to capture the data structure in the first two steps. Correctly setting either ρ or κ is a non-trivial task that is left to the user of these techniques. Furthermore, a kernel (most commonly an RBF) is often involved in the second step, adding an additional parameter (the kernel width/scale) to the user to determine. Expectedly, the optimal selection of these parameters plays a critical role in the overall success of the manifold learning process.

NOMAD departs drastically from this setup as no kernel selection nor nearest neighbor search are involved. Yet, the solution $\mathbf{Q}_*$ is effectively a kernel which is automatically learned from the data. Because $\mathbf{Q}_*$ is positive semidefinite it can be factorized as $\mathbf{Q}_* = \mathbf{Y}^\top \mathbf{Y}$, defining a feature map from $\mathbf{X}$ to $\mathbf{Y}$.

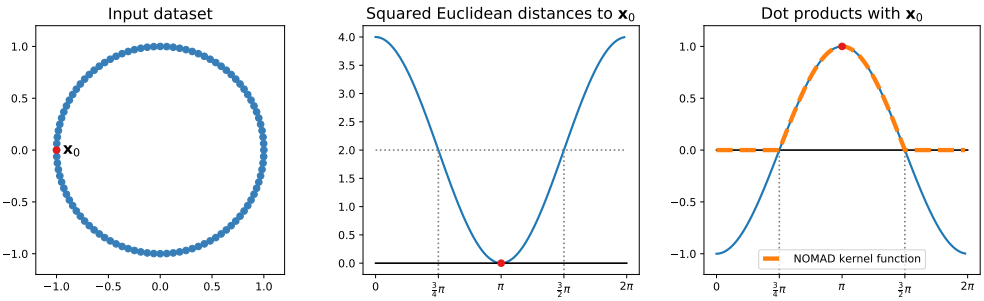


Figure 2: Correspondence between using a kernel/threshold in distance-space and nonnegativity of Gramian-based representations. In this toy example, the constraint $\mathbf{Q} \geq 0$ in NOMAD is equivalent to setting to zero distances that are greater than $\sqrt{2}$ (squared distances greater than 2). We use $\mathbf{x}_0$ as a reference but rotational symmetry makes this argument valid for all points in the dataset.

To illustrate intuitively the differences and similarities with prior work on manifold learning we use LLE (Roweis and Saul, 2000) as an example. LLE optimizes the cost function

$$\Phi(\mathbf{Y}) = \operatorname{Tr}\left((\mathbf{I}-\mathbf{W})^\top (\mathbf{I}-\mathbf{W})\left(\mathbf{Y}^\top \mathbf{Y}\right)\right), \tag{1}$$

where $\mathbf{W}$ is the adjacency matrix of a weighted nearest-neighbors graph. The key to finding a matrix $\mathbf{Y}$ that is locally isometric to $\mathbf{X}$, while unwrapping the data manifold, is to remove from $\mathbf{W}$ the connections between distant points $(\mathbf{X})_{:i}$ and $(\mathbf{X})_{:j}$. This is done with some technique to find nearest neighbors.

NOMAD also tries to align the output Gramian, $\mathbf{Q}$, to the input Gramian, $\mathbf{D}$, but discards distant data points differently. As negative entries in $\mathbf{D}$ cannot be matched because $\mathbf{Q}$ is nonnegative, the best option would be to set the corresponding element of $\mathbf{Q}$ to zero. This effectively discards pairs of input data points whose inner product is negative thus enforcing locality in the angular space (Cho and Saul, 2009), see Fig. 2. In fact, this argument can be taken further by noting that the constraint $\mathbf{Q1} = \mathbf{1}$ allows us to replace the Gramian $\mathbf{D}$ with the negative squared distance matrix,

$$-\tfrac{1}{2}\sum_{ij}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2\,(\mathbf{Q})_{ij} = -\sum_i (\mathbf{D})_{ii}\sum_j (\mathbf{Q})_{ij} + \operatorname{Tr}(\mathbf{DQ}) = -\operatorname{Tr}(\mathbf{D}) + \operatorname{Tr}(\mathbf{DQ}). \tag{2}$$

Finally, the constraint $\operatorname{Tr}(\mathbf{Q}) = K$ allows further control of the neighborhood size of NOMAD (modulating the actual width of its kernel function, see Fig. 2). Next, we develop further intuition about the manifold-learning capabilities of NOMAD by analyzing theoretically the dataset in Fig. 2.

As we mentioned before, the SDP formulation of Peng and Wei (2007) was developed as a clustering algorithm. Whether this method actually delivers a clustered solution depends on the geometry of the dataset. When the dataset consists of well-segregated clusters, the resulting $\mathbf{Q}_*$ has block diagonal structure. We empirically observe that, when the dataset is sampled from a regular manifold, the solution $\mathbf{Q}_*$ does not break down the dataset into artificial clusters and actually preserves the manifold structure (see Sec. 3). In a simple example, where the manifold exhibits a high degree of symmetry, we demonstrate analytically that this behavior occurs. The following sections are devoted to this task.

2.1. Analysis of NOMAD on a 2D ring dataset

We analyze the case in which the input data to NOMAD possess rotational symmetry, i.e., data are arranged uniformly on a ring, see Fig. 2. In this case, we can write the SDP as a linear program (LP) in the circular Fourier basis. This new representation allows to visualize that NOMAD lifts the data into a high-dimensional space, with K controlling its dimensionality.

In the example in Fig. 2, the entries of $\mathbf{D}$ can be described by $(\mathbf{D})_{ij} = \mathbf{x}_i{}^\top \mathbf{x}_j = \cos(\alpha_i - \alpha_j)$, where α_i, α_j are the angles of points $\mathbf{x}_i, \mathbf{x}_j$, respectively (Fig. 2). Since the points are uniformly distributed over the ring, $\mathbf{D}$ is a circulant matrix, i.e., $\cos(\alpha_i - \alpha_j) = \cos(\alpha_{i+k} - \alpha_{k+k})$. The solution $\mathbf{Q}_*$ to NOMAD is circulant too (Bachoc et al., 2012). Being circulant matrices, $\mathbf{D}$ and $\mathbf{Q}_*$ are diagonalized by the discrete Fourier transform (DFT), i.e.,

$$\mathbf{D} = \mathbf{F}\,\mathrm{diag}(\mathbf{d})\mathbf{F}^{\mathrm{H}}, \qquad \mathbf{Q}_* = \mathbf{F}\,\mathrm{diag}(\mathbf{q})\mathbf{F}^{\mathrm{H}}, \tag{3}$$

where $\mathbf{q}, \mathbf{d} \geq \mathbf{0}$ respectively are vectors containing the eigenvalues of $\mathbf{D}$ and $\mathbf{Q}_*$, $\mathbf{F}^{\mathrm{H}}$ is a Hermitian conjugate of $\mathbf{F}$, and $\mathbf{F} \in \mathbb{C}^{n\times n}$ is the unitary DFT matrix, with entries $(p, k = 0, \ldots, n-1)$

$$(\mathbf{F})_{pk} = \tfrac{1}{\sqrt{n}} \exp\left(-i2\pi p \tfrac{k}{n}\right). \tag{4}$$

Hence, and in accord with the constraint $\mathbf{Q}_*\mathbf{1} = \mathbf{1}$, we have that $(\mathbf{F})_{0:} = \frac{1}{\sqrt{n}}\mathbf{1}$ and $(\mathbf{q})_0 = 1$.

2.2. A linear program on the data manifold

We express the objective function and the constraints of NOMAD in terms of $\mathbf{d}$ and $\mathbf{q}$, i.e.,

$$\mathrm{Tr}\,(\mathbf{D}\mathbf{Q}_*) = \mathbf{d}^\top \mathbf{q}, \tag{5}$$

$$\mathrm{Tr}\,(\mathbf{Q}) = \mathbf{1}^\top \mathbf{q} = K, \tag{6}$$

$$(\mathbf{Q})_{kk'} = (\mathbf{F})_{k:}\,\mathrm{diag}(\mathbf{q})(\mathbf{F}^{\mathrm{H}})_{:k'} = \sum_{p=0}^{n-1} \tfrac{(\mathbf{q})_p}{n} \cos\left(2\pi p \tfrac{k'-k}{n}\right) \geq 0. \tag{7}$$

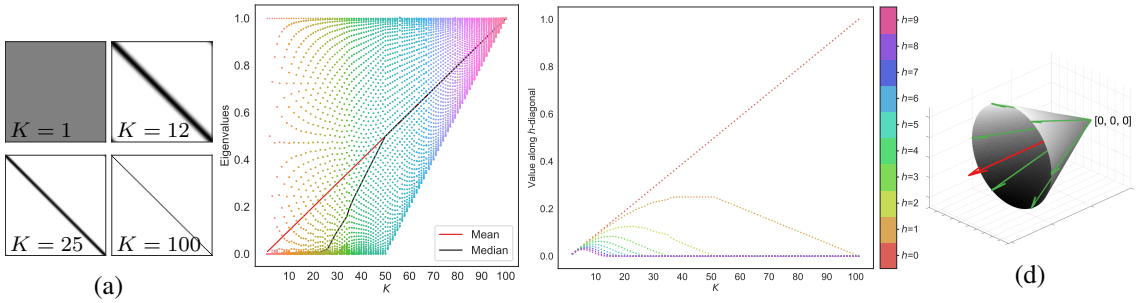
This reformulation allows us to rewrite NOMAD as a linear program

$$\max_{\mathbf{q}} \mathbf{d}^\top \mathbf{q} \quad \text{s.t.} \quad (\forall \tau)\ \mathbf{c}_\tau{}^\top \mathbf{q} \geq 0, \quad \mathbf{1}^\top \mathbf{q} = K, \quad \mathbf{q} \geq \mathbf{0}, \quad (\mathbf{q})_0 = 1, \tag{8}$$

where $(\mathbf{c}_\tau)_p = \frac{1}{n}\cos\left(2\pi p \frac{\tau}{n}\right)$.

Problem (8) sheds light on the inner workings of NOMAD. First, the constraint $\mathbf{1}^\top\mathbf{q} = K$ ensures that $\mathbf{q}$ does not grow to infinity and acts as a budget constraint. Let us assume for a moment that we remove the constraint $\mathbf{c}_\tau{}^\top\mathbf{q} \geq 0$ (the equivalent of $\mathbf{Q} \geq \mathbf{0}$). Then, the program will try to set to K the entry of $\mathbf{q}$ corresponding to the largest eigenvalue of $\mathbf{d}$; this $\mathbf{q}$ will violate as K gets bigger the removed constraint (since $(\mathbf{c}_\tau)_p$ is a sinusoid). Then the effect of this constraint is to spread the allocated budget among several eigenvalues (instead of just the largest). The experiment in Fig. 3 confirms this: the number of active eigenvalues of $\mathbf{Q}_*$ grows with K. We can interpret this as increasing the intrinsic dimensionality of the problem in such a way that only local interactions are considered.

Interpretation of K. The circulant property of $\mathbf{Q}_*$ for the 2D ring sheds further light on the meaning of K. In Fig. 3(c), we observe that the number of significant elements in each of $\mathbf{Q}_*$ is $\lceil n/K \rceil$.



(a) (b) (c) (d)

Figure 3: Evolution of the NOMAD solution for the 2D ring dataset (with 100 points, see Fig. 2) with increasing parameter K. (a) As K increases, the solution, $\mathbf{Q}_*$, concentrates more and more towards the diagonal. (b) As K increases, the number of active eigenvalues in the solution, $\mathbf{Q}_*$, grows resulting in the more uniform distribution of eigenvalues and greater mean/median (notice that the mean being linear comes from the trace constraint). (c) We define the h-diagonal of $\mathbf{Q}_*$ as the entries (i, j) for which $i - j = h$. As $\mathbf{Q}_*$ is a circulant matrix, each h-diagonal contains a single repeated value. We plot these values, assigning a different color to each h. The effect of the scaling constraint $\mathrm{Tr}\,(\mathbf{Q}) = K$ becomes evident: when one h-diagonal becomes inactive, all remaining h'-diagonals need to be upscaled. (d) The eigenvectors of $\mathbf{Q}_*$ form a high-dimensional cone (cartoon representation, cone axis in red and eigenvectors in green).

Thus, we can interpret K as a parameter that effectively sets the size of the local neighborhood on the manifold. In standard manifold learning methods this size is set by a combination of the number of nearest neighbors and the shape and scale of the kernel function. In NOMAD, all these variables are incorporated into a single parameter and balanced with the help of the remaining problem constraints.

In general, for non-symmetric and irregularly sampled manifolds, K is chosen to capture the manifold underlying the dataset: the neighborhood size needs to be small enough to capture the desired manifold features, but big enough to avoid capturing unwanted structure (e.g., noise). If sampling density differs in different areas, the size will adjust locally as needed.

2.3. Lifting the ring to a high-dimensional cone

Here, we show that NOMAD effectively embeds the data manifold into a space where its structure, i.e., rotational symmetry, is preserved. We now make use of the half-wave symmetry in $\mathbf{Q}_*$, noting that they can be fully represented with only one half of the Fourier basis. We can then decompose it with the real Fourier basis

$$\mathbf{Q}_* = \tilde{\mathbf{F}}\,\mathrm{diag}(\tilde{\mathbf{q}})\tilde{\mathbf{F}}^\top, \tag{9}$$

where $\tilde{\mathbf{q}} = [(\mathbf{q})_0, (\mathbf{q})_1, (\mathbf{q})_1, \ldots, (\mathbf{q})_{n-1}, (\mathbf{q})_{n-1}]^\top$ and $\tilde{\mathbf{F}} \in \mathbb{R}^{n\times n}$ has entries $(p, k = 0, \ldots, n-1)$

$$(\tilde{\mathbf{F}})_{pk} = \begin{cases} \frac{2}{\sqrt{n}} \cos\left(2\pi p \frac{k}{n}\right) & \text{if } k \text{ is even,} \\ \frac{2}{\sqrt{n}} \sin\left(2\pi p \frac{k-1}{n}\right) & \text{if } k \text{ is odd.} \end{cases} \tag{10}$$

Let $\tilde{\mathbf{Y}} = \mathrm{diag}(\mathbf{q})^{1/2}\tilde{\mathbf{F}}^\top$. Notice that $\langle \tilde{\mathbf{Y}}_{:i}/\left\|\tilde{\mathbf{Y}}_{:i}\right\|_F, \tilde{\mathbf{F}}_{:0}\rangle = \langle \tilde{\mathbf{F}}_{:i}, \tilde{\mathbf{F}}_{:0}\rangle = \frac{4}{n}$, meaning that the vectors $\tilde{\mathbf{Y}}_{:i}$ are the extreme rays of a right circular cone with the eigenvector $\tilde{\mathbf{F}}_{:0} = \frac{2}{\sqrt{n}}[1, 0, \ldots, 0]^\top$ as its symmetry axis, see Fig. 3(d). Thus, we can interpret the solution to NOMAD as lifting the 2D ring structure into a cone. As mentioned before, this cone is high-dimensional, with as many directions as needed to preserve the nonnegativity of $\mathbf{Q}$.

We identify the rank of the solution $\mathbf{Q}$ with the number of active eigenvalues. The bigger the K, the higher the rank. The constraint $\mathbf{Q1} = \mathbf{1}$ in NOMAD leads to a fanning-out effect in the data representation. Intuitively, this fan-out effect is key to the disentanglement of datasets with complex topologies. Spin-model-inspired SDPs for community detection (Javanmard et al., 2015) achieve a similar fanning-out by dropping the constraint $\mathbf{Q1} = \mathbf{1}$ and adding the related term $-\gamma\mathbf{1}^\top\mathbf{Q1}$ to the objective function.

With the LP framework and the geometric picture in place, we can begin to understand how the solution evolves as the parameter K increases from 1 to n. At $K = 1$, only the eigenvalue $(\mathbf{q})_0$ is active and every vector $(\tilde{\mathbf{Y}})_{:i}$ is the same with each entry equal to $1/n$. When K slightly above 1, the eigenvalue $(\mathbf{q})_1$ becomes active (nonzero), introducing the first nontrivial Fourier component. Geometrically, the vectors $\{(\tilde{\mathbf{Y}})_{:i}\}$ now open up into a narrow cone. As K increases, the cone widens and, at some point, the angle between two of the vectors reaches $\pi/2$ (this activates the nonnegativity constraint in Eq. (7)). Further increase of K necessitates use of a larger number of Fourier modes. Finally, at $K = n$ all modes are active and all vectors $\{(\tilde{\mathbf{Y}})_{:i}\}$ become orthogonal to each other. Fig. 3(b) depicts the progression with K of the number of active modes.

Summary. Previous studies (Kulis et al., 2007; Peng and Wei, 2007; Awasthi et al., 2015), focus solely on cases where NOMAD exhibits K-means-like solutions (i.e., hard-clustering). Sec. 2 provides a characterization of the NOMAD solutions on a simple example with a high degree of symmetry, showing that they are drastically different from K-means. These solutions connect neighboring points, with the neighborhood size determined by K. These neighborhoods overlap, as they would in soft-clustering, in a way that preserves global features of the manifold, including its symmetry. This is a feature sought after by manifold learning methods and help place NOMAD among reliable manifold analysis techniques.

3. Analyzing data manifolds with NOMAD: Experimental results

In the previous section, we showed that NOMAD recovers the data manifold in an idealized 2D ring dataset. Here, we extend this observation numerically to more complex datasets for which analytical form of the transformation that diagonalizes $\mathbf{Q}_*$ (nor $\mathbf{D}$) is not known, see figs. 1, 5 and 7. We visualize the solution $\mathbf{Q}_*$ by embedding it in a low-dimensional space. While our goal is not dimensionality reduction, we learn the data manifold with NOMAD, and use standard spectral dimensionality reduction to visualize the results.

Recovering multiple manifolds. K-means cannot effectively recover multiple distinct manifolds (although in some very particular cases, with well separated and linearly separable manifolds, it may group the points correctly). Interestingly, NOMAD does not inherit this limitation. Of course, if we set the NOMAD parameter K to the number of manifolds that we want to recover, there is no hope in the general case to obtain a result substantially better than the one obtained with Lloyd's algorithm (Lloyd, 1982). However, setting the NOMAD parameter K to be higher than the number of manifolds leads to a correct identification and characterization of their structures. Note that

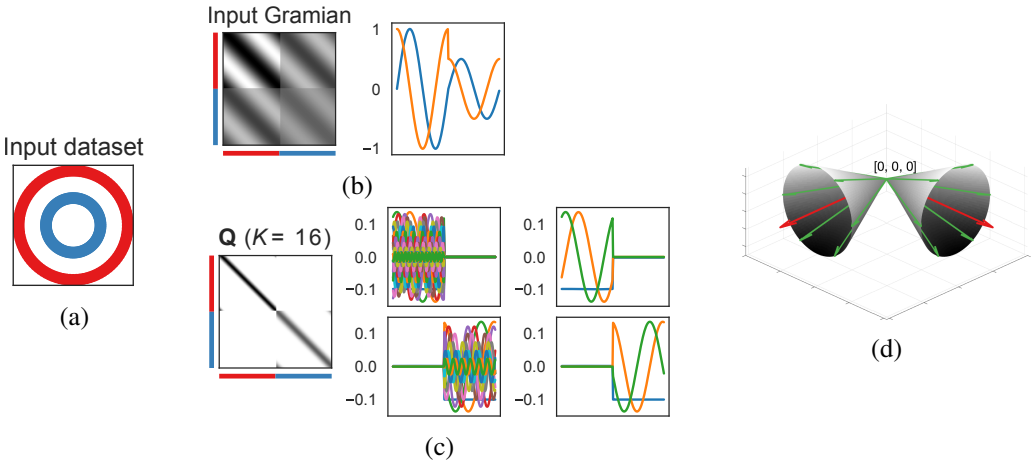


Figure 4: Solution of NOMAD on the dataset consisting of two 2D rings. (a) Two-ring dataset. (b) Input Gramian, $\mathbf{D}$, and its two eigenvectors ($\mathbf{D}$ has rank 2). Note that the eigenvectors of $\mathbf{D}$ do not segregate the rings. (c) The solution, $\mathbf{Q}$, of NOMAD contains two sets of eigenvectors with disjoint support: one set describing the points in each ring (we show all eigenvectors and a detail on the first 3 within each set). (d) The eigenvectors of $\mathbf{Q}$ form two orthogonal high-dimensional cones: one cone for each ring (cartoon representation, cone axis in red and eigenvectors in green). Notice how these cones become linearly-separable.

setting a similarly large K would not help K-means, as it is designed to partition the data, thus breaking each manifold into several pieces.

An example with two rings is presented in Fig. 4. We can expect that, as the single ring in Sec. 2 is described by Fourier modes, NOMAD describes two rings with two sets of Fourier modes with disjoint support; the solution is now arranged as two orthogonal high-dimensional cones, see Fig. 4(d). In a sense, the manifold learning problem is already solved, as there are two circulant submatrices, one for each manifold, with no interactions between them. If the user desires a hard assignment of points to manifolds, we can simply consider $\mathbf{Q}_*$ as the adjacency matrix of a weighted graph and compute its connected components.

Discussion of the experimental results. To demonstrate the manifold-learning capabilities of the NOMAD, we present several examples, both synthetic and real-world. The trefoil knot in Fig. 1(a) is a 1D manifold in 3D; it is the simplest example of a nontrivial knot, meaning that it is not possible to "untie" it in three dimensions without cutting it. However, the manifold learning procedure in Sec. 3 learns a closed 1D manifold. We also present examples using real-world high-dimensional datasets, recovering in every case structures of interest, see Fig. 6. In figs. 6(a) to 6(c), NOMAD respectively uncovers the camera rotation, the orientation of the lighting source, and specific handwriting features.

To demonstrate the multi-manifold learning and manifold-disentangling capabilities of NOMAD, we use several standard synthetic datasets, see figs. 1(b), 4 and 5. In all of these examples, NOMAD is able to disentangle clusters that are not linearly separable. We also present results for a real-world dataset (Fig. 7) which is similar to the one in Fig. 6(a) but with two objects. NOMAD

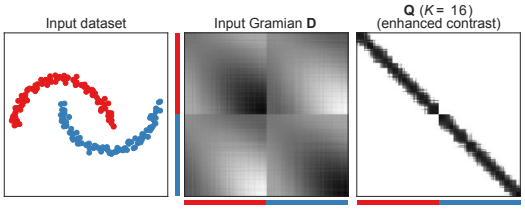
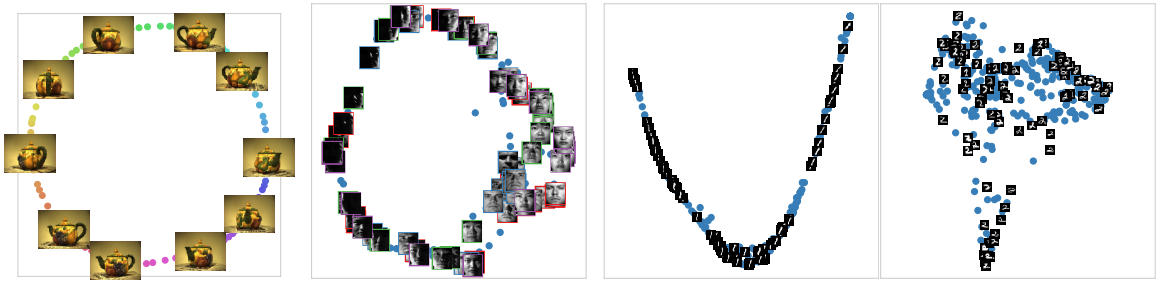


Figure 5: Learning multiple manifolds with NOMAD. The points are arranged in two semicircular manifolds and contaminated with Gaussian noise. Although the manifolds are linearly non-separable, NOMAD correctly finds two submatrices, one for each manifold (for visual clarity, we enhance the contrast of $\mathbf{Q}$).



(a) Teapots ($K = 20$) (b) Yale Faces ($K = 16$) (c) MNIST digits ($K = 16$)

Figure 6: Finding two-dimensional embeddings with NOMAD. (a) 100 images obtained by viewing a teapot from different angles in a plane. The input vectors size is 23028 (76×101 pixels, 3 color channels). The manifold uncovers the change in orientation. (b) 256 images from 4 different subjects (each subject is marked with a different color in the figure), obtained by changing the position of the illumination source. The input vectors size is 32256 (192×168 pixels). The manifold uncovers the change in illumination (from frontal, to half-illuminated, to dark faces, and back). (c) 500 images handwritten instances of the same digit. The input vectors size is 784 (28×28 pixels). On the left and on the right, images of the digits 1 and 2, respectively. The manifold of 1s uncovers their orientation, while the manifold of 2s parameterizes features like size, slant, and line thickness. Details are better perceived by zooming on the plots.

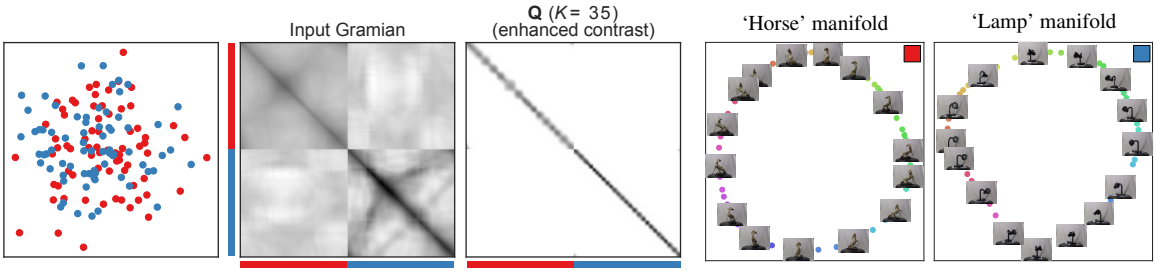
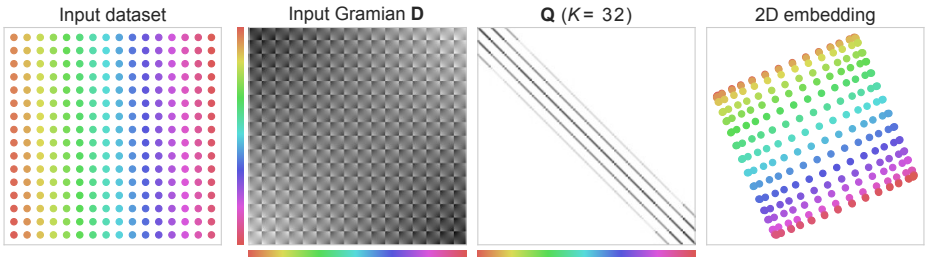
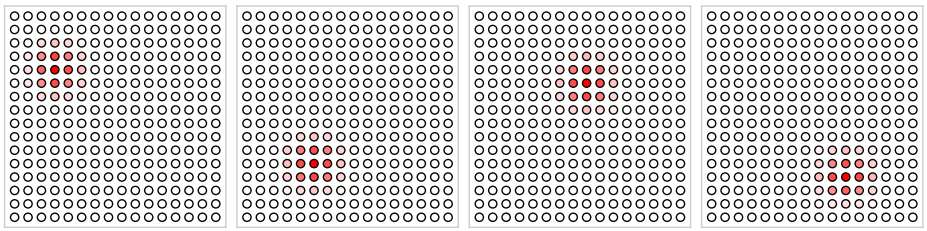


Figure 7: 144 images obtained by viewing a lamp and a horse figurine from different angles in a plane. The input vectors size is 589824 (384×512 pixels, 3 color channels). We plot the input data using a 2D spectral embedding (the points corresponding to each object are colored differently). NOMAD correctly finds two submatrices, one for each manifold (for visual clarity, we enhance the contrast of $\mathbf{Q}$); furthermore, NOMAD recovers closed manifolds.



(a)



(b)

Figure 8: Learning a 2D manifold embedded in a 10-dimensional ambient space; the first two dimensions are regular samples on a 2D grid (shown on the left) and the remaining ones are Gaussian noise. (a) NOMAD recovers the right 2D structure (of course, some distortion is expected along the edges). (b) We show a few columns of $\mathbf{Q}$ on top of the data itself: the red level indicates the value of the corresponding entry in $\mathbf{Q}$ (red maps to high values, white maps to a zero). NOMAD effectively tiles the dataset with a collections of overlapping local neighborhoods centered at each data point. These patches contain all the information necessary to reconstruct the intrinsic manifold geometry.

recovers two closed manifolds, each of which containing the viewpoints of one object. The structure of the solution is similar to the one in Fig. 4(d).

Finally, we include an example in which NOMAD captures the structure of a 2D manifold living in a 10-dimensional space, see Fig. 8. NOMAD assigns a local patch to each data point (non-zero values for neighboring points, zeros elsewhere). These local patches tile the manifold with overlap (as in soft-clustering), allowing to recover its grid structure. Such tiling takes place in all of the examples included in the paper.

3.1. Manifold disentangling with multi-layer NOMAD

The recursive application of NOMAD, with successively decreasing values of K, enhances its manifold-disentangling capabilities. The pseudocode is as follows:

$\mathbf{D}_1 \leftarrow \mathbf{X}^\top \mathbf{X}$;
for $l = 1, 2, \ldots$ **do**
  Choose K_l (for all $l > 1$ we require $K_l \leq K_{l-1}$);
  Find the solution $\mathbf{Q}_l$ of NOMAD with input matrix $\mathbf{D}_l$ and parameter K_l;
  $\mathbf{D}_{l+1} \leftarrow \mathbf{Q}_l$;
return $\{\mathbf{Q}_l\}_{l=1,2,\ldots}$

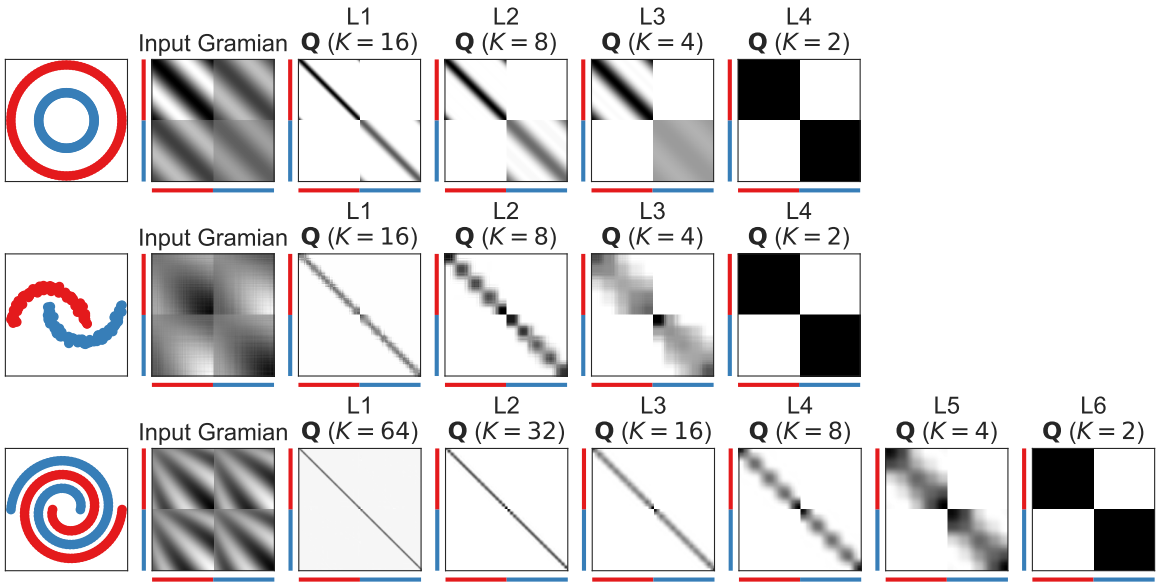


Figure 9: Results of recursive NOMAD application (multi-layer NOMAD). For each example, we show matrices $\mathbf{Q}_*$ computed by the successive application of the algorithm. Multi-layer NOMAD untangles these linearly non-separable manifolds and, in the final layer, assigns each manifold to one cluster.

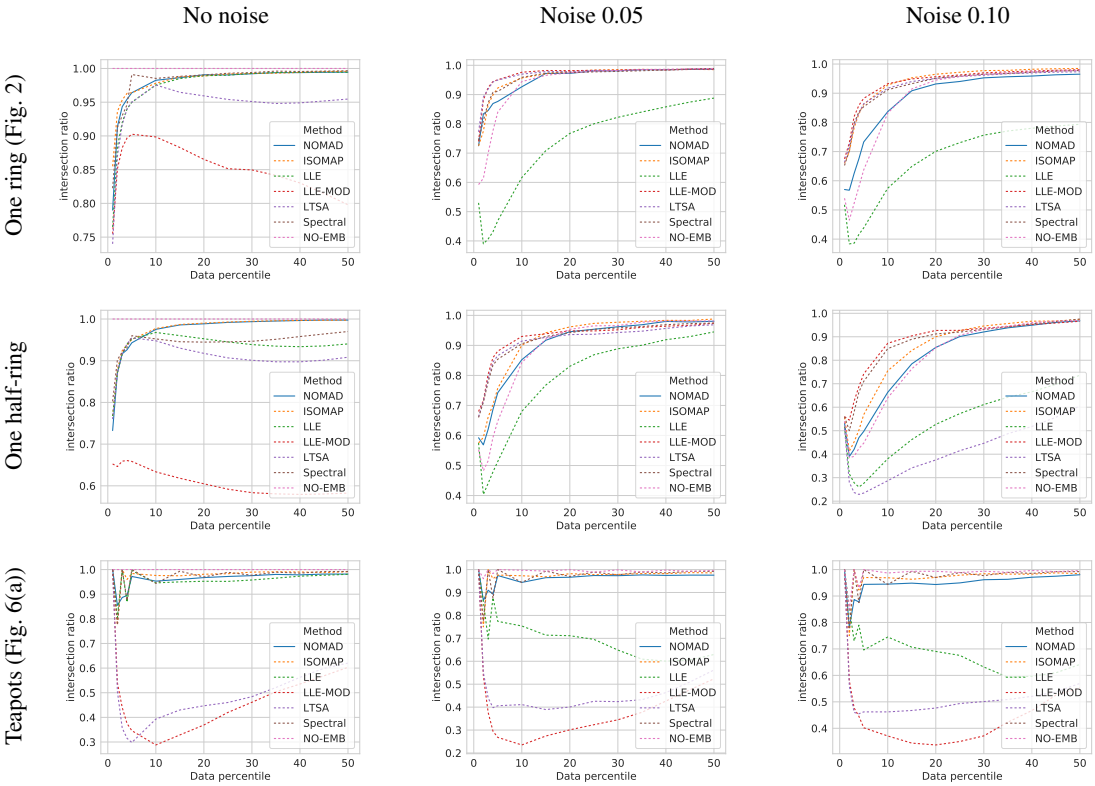
In Fig. 9, we present the evolution of successive matrices $\mathbf{Q}_l$. In all of these examples, multi-layer NOMAD is able to correctly identify clusters that are not linearly separable, something unattainable with single-layer NOMAD or with K-means clustering. Interestingly, we find that the manifolds are already segregated after one application of NOMAD in the direction of the leading eigenvectors of $\mathbf{Q}$ (see Fig. 4). The rest of the NOMAD layers little-by-little sieve out the (unwanted) smaller eigenvalues in an unsupervised fashion.

To turn this algorithm into a general data-analysis tool, we need an automated selection of the values $\{K_l\}$ which is a non-trivial task in general. Additional results, using different sequences $\{K_l\}$, can be found in Appendix C. Further research is needed to develop such algorithm and fully understand multi-layer NOMAD's interesting behavior.

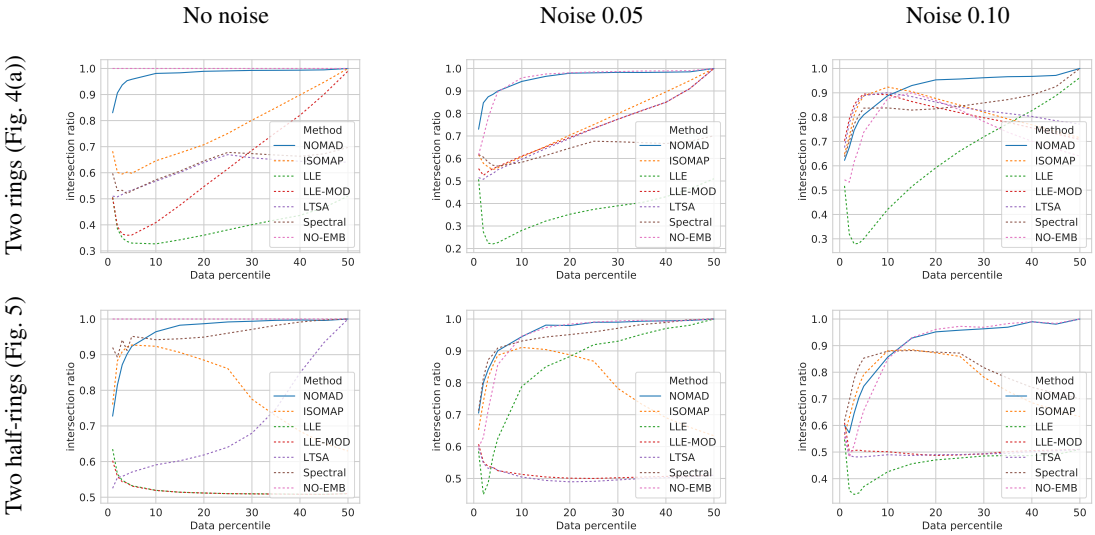
3.2. Geodesic-distance preservation: NOMAD versus existing manifold learning techniques

In order to compare different methods for manifold discovery, we need to agree upon appropriate metrics, which itself is an active area of research (e.g., Zhang et al., 2012). In particular, we want a metric that allows fair comparison among outputs of methods with very different objectives. Since our method is not explicitly geared towards dimension reduction, or towards variance maximization, we prefer metrics that emphasize the preservation of intrinsic structure of the manifold. In particular, we hope to preserve the ordering of intrinsic distances along the manifold, something that guarantees that the neighborhood structure remain similar.

Concretely, (1) we compute N nearest neighbors for each dataset point and build a weighted graph with these distances as edges; (2) we use this graph to compute geodesic distances using



(a) One manifold: NOMAD is among the best-in-class methods.



(b) Two manifolds: NOMAD outperforms all considered methods.

Figure 10: Comparison of the robustness of geodesic distances to the addition of noise for different manifold learning methods. The description of the experimental protocol is in Sec. 3.2. The method termed NO-EMB directly computes distances on the noisy data.

Dijkstra's algorithm; (3) we finally sort the distances in increasing order. We consider this ordering our ground truth.

We then add noise to the each point in the dataset. Noise was added by creating $5d$ additional dimensions that contain Gaussian noise with standard deviation 0.05 and 0.10. Using the noisy dataset, we run the geodesic-distance-sorting process on the embeddings produced by different manifold learning algorithms using N nearest neighbors. For NOMAD, instead of resorting to nearest neighbors, we set $K = n/N$ and use the non-zero entries in $\mathbf{Q}$ to determine the graph connectivity. As NOMAD yields a similarity matrix $\mathbf{Q}$, we derive distances from it with the formula $(\mathbf{Q})_{ii} + (\mathbf{Q})_{jj} - 2(\mathbf{Q})_{ij}$. Using this weighted graph, we compute and sort the geodesic distances.

For our distance-preservation measure, we use a bullseye score: for each method, we count the fraction of points in the top p percentile of distances that are also present in the top p percentile of ground truth distances.

As seen in Fig. 10, when the data is sampled from a single manifold, NOMAD performs very well, on par with the best algorithms included in our comparison. However, in the two-manifolds case, NOMAD clearly outperforms all other methods, nearly matching the performance of direct distance computations on the noisy data.

4. Heuristic non-convex solvers for large-scale NOMAD

Standard SDPs involve $O(n^2)$ variables and their resulting time complexity is often $O(n^3)$. Consequently, standard solvers (O'Donoghue et al., 2016a) will struggle with large datasets. NOMAD lends itself to a fast and big-data-friendly implementation (Kulis et al., 2007). This is done by posing a related problem

$$\max_{\mathbf{Y}\in\mathbb{R}^{r\times n}} \operatorname{Tr}\left(\mathbf{D}\mathbf{Y}^\top\mathbf{Y}\right) \;\text{ s.t. }\; \operatorname{Tr}\left(\mathbf{Y}^\top\mathbf{Y}\right) = K,\; \mathbf{Y}^\top\mathbf{Y}\mathbf{1} = \mathbf{1},\; \mathbf{Y} \geq \mathbf{0}. \tag{11}$$

In this new problem, we have forgone convexity in exchange of reducing the number of unknowns from $O(n^2)$ to rn. For example, Kulis et al. (2007) set $r = K$. The problematic constraint $\mathbf{Y}^\top\mathbf{Y} \geq \mathbf{0}$, involving $O(n^2)$ terms, has been replaced by the much stronger but easier to enforce $\mathbf{Y} \geq \mathbf{0}$. The speed gain is shown in Fig. 15. See Appendix E for a description of the algorithm.

However, strictly speaking, the new constraint is equivalent to the old one only if $\mathbf{Q}$ is completely positive. An $n \times n$ matrix $\mathbf{A}$ is called completely positive (CP) if there exists $\mathbf{B} \geq \mathbf{0}$ such that $\mathbf{A} = \mathbf{B}^\top\mathbf{B}$. The least possible number of rows of $\mathbf{B}$ is called the cp-rank of $\mathbf{A}$. Whereas matrix $\mathbf{A}$ is doubly nonnegative (DN), i.e. $\mathbf{A} \geq \mathbf{0}$ and $\mathbf{A} \succeq 0$, not every DN matrix (with $n > 4$) is CP (Maxfield and Minc, 1962).

We are thus interested in two questions. First, is the solution $\mathbf{Q}_*$ to NOMAD completely positive? Answering this question in the affirmative would allow for theoretically sound and fast implementations of NOMAD. Whereas the set of CP matrices forms a convex cone, the problems of determining whether a matrix is inside the set and of projecting a matrix into the set are NP-hard leading us to the second question: What is the cp-rank of $\mathbf{Q}_*$? This issue is critical because it determines the number of unknowns. For example, if $\text{cp-rank}(\mathbf{Q}_*) \leq K$, (11) would be easier to solve. These questions are difficult only when NOMAD produces a soft-clustering $\mathbf{Q}_*$, as in all of the examples in this paper. Indeed, it is not hard to prove that, whenever NOMAD produces a hard-clustering $\mathbf{Q}_*$, $\mathbf{Q}_*$ is CP (see Awasthi et al., 2015, for such conditions).

Let us now go back to the example in Sec. 2 (points arranged regularly on a ring). For this example, we can establish a simple sufficient condition on K, for $\mathbf{Q}_*$ to be CP. Recall that if $\mathbf{D}$ is

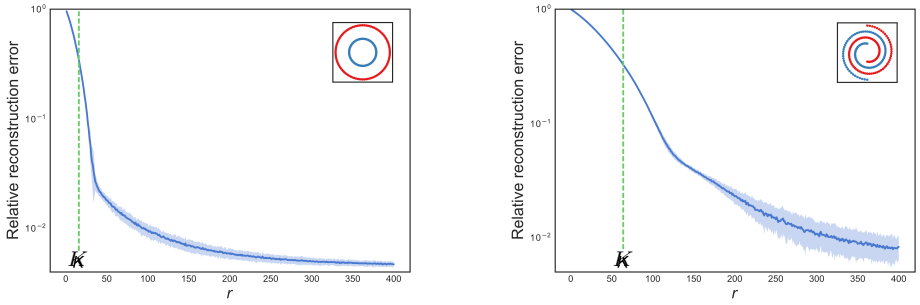


Figure 11: We empirically study the cp-rank of $\mathbf{Q}_*$. As a proxy of the exact nonnegative decomposition, we compute the rank-r symmetric NMF $\mathbf{Q}_* \approx {\mathbf{Y}_+}^\top \mathbf{Y}_+$ for different values of r. We show the mean plus/minus two standard deviations of the relative error $\left\|\mathbf{Q}_* - {\mathbf{Y}_+}^\top \mathbf{Y}_+\right\|_F / \left\|\mathbf{Q}_*\right\|_F$ computed from 50 different SNMFs for each r (their differences stem from the random initialization). Both datasets have 200 points. Clearly, setting $r = K$ is not enough to properly reconstruct $\mathbf{Q}_*$.

circulant, $\mathbf{Q}_*$ is circulant (Bachoc et al., 2012). In Proposition 2 of Appendix B, we prove that if the solution $\mathbf{Q}_*$ to NOMAD is a circulant matrix, then it is CP for every $K \leq 3/2$ or $K \geq \frac{n}{2}$. Naturally, more theory is needed to shed light onto this problem in general scenarios as it is unclear whether similar results exist.

Complementarily, we have studied the questions raised in this section from an experimental viewpoint. We use the symmetric nonnegative matrix factorization (SNMF) of $\mathbf{Q}_*$, see Appendix D, as a proxy for checking whether $\mathbf{Q}_*$ is CP. The rationale is that if the approximation with SNMF is very tight, it is highly likely that $\mathbf{Q}_*$ is CP. These experiments are presented in Fig. 11. We found that, with a properly chosen rank r, SNMF can indeed accurately approximate $\mathbf{Q}_*$. However, setting $r = K$ is in general not enough and leads to a poor reconstruction. These two facts support the idea that $\mathbf{Q}_*$ is CP, but has a cp-rank much higher than K.

Our experiments with the non-convex algorithm in Appendix E lead to similar conclusions as those with SNMF, see Fig. 12. Setting $r = K$, leads to a poor approximation of $\mathbf{Q}_*$ and, as observed by Kulis et al. (2007), to hard-clustering. Setting $r \gg K$ leads to much improved reconstructions, at the expense of speed.

5. A fast and convex algorithm for NOMAD

The Burer-Monteiro solver forgoes convexity in favor of speed. However, as discussed in the previous section, this conversion carries theoretical and practical difficulties that are not easily overcome. In this section, we propose an algorithm for NOMAD that is fast and yet convex.

5.1. Augmented Lagrangian formulation

First, we redefine the variables in NOMAD by setting $\mathbf{P} = \mathbf{Q} - \mathbf{E}_n$, where $\mathbf{E}_n = \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. Then,

$$\max_{\mathbf{P}} \operatorname{Tr}(\mathbf{D}\mathbf{P}) \quad \text{s.t.} \quad \mathbf{P}\mathbf{1} = \mathbf{0}, \quad \operatorname{Tr}(\mathbf{P}) = K - 1, \quad \mathbf{P} \succeq \mathbf{0}, \quad \mathbf{P} + \mathbf{E}_n \geq \mathbf{0}. \tag{12}$$

As usual in the optimization literature, we handle this constraint with an augmented Lagrangian method. The augmented Lagrangian of Problem (12) with respect to the constraint $\mathbf{P} + \mathbf{E}_n \geq \mathbf{0}$ is

$$g(\mathbf{P}, \mathbf{\Gamma}) = -\operatorname{Tr}(\mathbf{D}\mathbf{P}) + \operatorname{Tr}(\mathbf{\Gamma}(\mathbf{P} + \mathbf{E}_n)) + \tfrac{\gamma}{2}\left\|[\mathbf{P} + \mathbf{E}_n]_-\right\|_F^2, \tag{13}$$

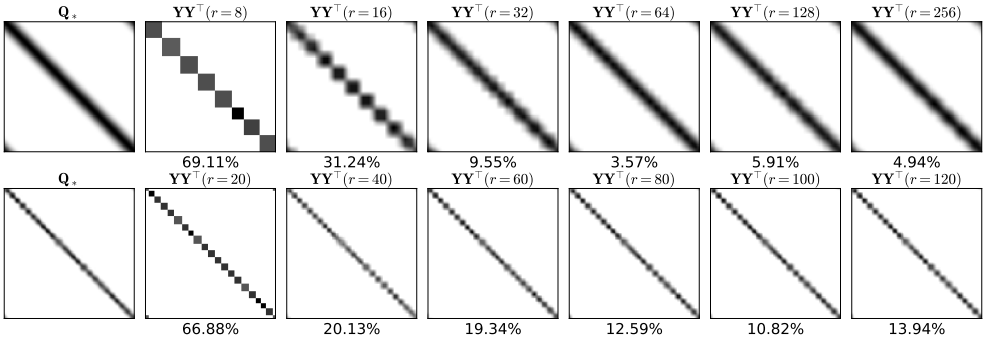


Figure 12: Comparison of the results obtained with a standard SDP solver (first column) and with a low-rank non-convex approach (remaining columns, r denotes the rank of the obtained solution). **(Top)** Dataset in Fig. 4; we set $K = 8$ in all cases. **(Bottom)** Dataset in Fig. 6(a); we set $K = 20$ in all cases. In each case, we also display the relative error between the matrix $\mathbf{Y}^\top\mathbf{Y}$ and $\mathbf{Q}_*$. Interestingly, setting $r = K$ produces hard clustering solutions (see the block diagonal structure of the matrices on the second column), while increasing r produces "softer" solutions. This suggests that the cp-rank of $\mathbf{Q}_*$ is (much) greater than K.

where $\mathbf{\Gamma} \geq \mathbf{0}$ is the associated Lagrange multiplier, and $[\cdot]_- = \min(\cdot, 0)$ is the projection operator onto the negative orthant. We can then pose Problem (12) as

$$\min_{\mathbf{P}} \max_{\mathbf{\Gamma}\geq 0} g(\mathbf{P}, \mathbf{\Gamma}) \quad \text{s.t.} \quad \mathbf{P1} = \mathbf{0}, \quad \operatorname{Tr}(\mathbf{P}) = K - 1, \quad \mathbf{P} \succeq \mathbf{0}. \tag{14}$$

We solve it using the method of multipliers, i.e.,

$$\mathbf{P}_{t+1} = \operatorname*{argmin}_{\mathbf{P}} g(\mathbf{P}, \mathbf{\Gamma}_t) \quad \text{s.t.} \quad \mathbf{P1} = \mathbf{0}, \quad \operatorname{Tr}(\mathbf{P}) = K - 1, \quad \mathbf{P} \succeq \mathbf{0}, \tag{15a}$$

$$\mathbf{\Gamma}_{t+1} = [\mathbf{\Gamma}_t + \tau(\mathbf{P}_{t+1} + \mathbf{E}_n)]_- . \tag{15b}$$

5.2. A conditional gradient method for SDPs with an orthogonality constraint

In this section, we introduce a very efficient algorithm to solve

$$\max_{\mathbf{Z}} f(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z} \succeq 0,\ \operatorname{Tr}(\mathbf{Z}) = s,\ \mathbf{Zb} = \mathbf{0}. \tag{16}$$

of which Problem (15a) is an instance.

To this end we modify an algorithm to efficiently solve the SDP (Hazan, 2008)

$$\max_{\mathbf{Z}} f(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z} \succeq 0,\ \operatorname{Tr}(\mathbf{Z}) = s, \tag{17}$$

where function f is differentiable and concave. The iterative algorithm consists, at each iteration $t = 0 \ldots$, of the following steps:

1. Let $\mathbf{v}_t$ be the largest algebraic eigenvector of $\nabla f(\mathbf{Z}_t)$.
2. $\mathbf{Z}_{t+1} = (1 - \alpha)\mathbf{Z}_t + \alpha s \mathbf{v}_t \mathbf{v}_t{}^\top$ with $\alpha = 2/(t + 2)$.

Algorithm 1: Conditional gradient algorithm for SDPs with an orthogonality constraint

input : function f to minimize, scale parameter s.
output : solution $\mathbf{Z}_{t+1} \in \mathcal{P}_s$ to Problem (17).

Initialize $\mathbf{Z}_0 = \mathbf{0}$;
for $t = 0, \ldots, \infty$ **do**
  Let $\mathbf{v}$ be the largest algebraic eigenvector of $\nabla f(\mathbf{Z})$ such that $\mathbf{v}^\top \mathbf{b} = 0$;
  $\alpha \leftarrow 2/(t+2)$;
  $\mathbf{Z}_{t+1} \leftarrow (1-\alpha)\mathbf{Z}_t + \alpha s \mathbf{v}\mathbf{v}^\top$;
  if *converged* **then** break ;

This algorithm is an instance of the Frank-Wolfe/conditional-gradient algorithm (Frank and Wolfe, 1956). As such it provides a solution without performing any projections. First, $\mathbf{Z}_{t+1}$ is a non-negative linear combination of two positive semidefinite matrices, and is thus positive semidefinite itself. Second, the iterations maintain the invariant $\operatorname{Tr}(\mathbf{Z}_t) = s$ as $\operatorname{Tr}(\mathbf{Z}_{t+1}) = (1-\alpha)\operatorname{Tr}(\mathbf{Z}_t) + \alpha s \operatorname{Tr}\left(\mathbf{v}_t \mathbf{v}_t{}^\top\right) = (1-\alpha)\operatorname{Tr}(\mathbf{Z}_t) + \alpha s$.

We now show how to extend this algorithm to handle an orthogonality constraint. Let $\mathcal{P}_s$ be the convex cone of positive semidefinite matrices with trace s that are orthogonal to a given vector $\mathbf{b}$, i.e.,

$$\mathcal{P}_s = \{\mathbf{Z} \succeq 0,\ \operatorname{Tr}(\mathbf{Z}) = s,\ \mathbf{Z}\mathbf{b} = \mathbf{0}\}. \tag{18}$$

Notice that setting $\mathbf{b} = \mathbf{1}$ yields the constraints of Problem (15a). We seek to solve

$$\max_{\mathbf{Z}} f(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{Z} \in \mathcal{P}_s. \tag{19}$$

Fortunately, we can push the constraint $\mathbf{Z}\mathbf{b} = \mathbf{0}$ into the eigenvector computation. We begin by noticing that the final solution is a weighted sum of the matrices $\mathbf{v}_t \mathbf{v}_t{}^\top$. It then suffices to require that, for every t, $\mathbf{v}_t \mathbf{v}_t{}^\top \mathbf{b} = \mathbf{0}$, which reduces to $\mathbf{v}_t{}^\top \mathbf{b} = \mathbf{0}$. This naturally yields a new iterative method, summarized in Alg. 1. This algorithm has the same performance guarantee as Hazan's (2008), given by the following proposition, which we prove in Appendix F.

Proposition 1 *Let $\mathbf{X}, \mathbf{Z} \in \mathcal{P}_s$ and $\mathbf{Y} = \mathbf{X} + \alpha(\mathbf{Z} - \mathbf{X})$ and $\alpha \in \mathbb{R}$. The curvature constant of f is*

$$C_f := \sup_{\mathbf{X},\mathbf{Z},\alpha} \tfrac{1}{\alpha^2}[f(\mathbf{X}) - f(\mathbf{Y}) + (\mathbf{Y} - \mathbf{X}) \bullet \nabla f(\mathbf{X})]. \tag{20}$$

Let $\mathbf{Z}^\star$ be the solution to Problem (19). The iterates $\mathbf{Z}_t$ of Alg. 1 satisfy for all $t > 1$

$$f(\mathbf{Z}^\star) - f(\mathbf{Z}_t) \leq \tfrac{8C_f}{t+2}. \tag{21}$$

5.3. A conditional gradient algorithm for NOMAD

Alg. 2 summarizes the proposed method of multipliers, see iterations (15a) and (15b), to solve Problem (12). The inner problem (15a) is solved using Alg. 1. A few remarks are in order:

- When using the method of multipliers, it is often not necessary (nor desirable) to solve the inner problem to a high precision (Goldstein and Osher, 2009). In our implementation we set $N_{\text{inner}} = 10$.

Algorithm 2: Conditional gradient algorithm for NOMAD

input : matrix $\mathbf{D}$, scale parameter k.
output : solution $\mathbf{Q}$ to NOMAD.

Initialize $\mathbf{P}_0 = \mathbf{0}$; $\quad \mathbf{\Gamma} \leftarrow \mathbf{0}$; $\quad \gamma = 1$;
for $t = 1, \ldots, \infty$ **do**
  for $t_{inner} = 1, \ldots, N_{inner}$ **do**
    Let $\nabla g(\mathbf{P}, \mathbf{\Gamma}) = -\mathbf{D} + \mathbf{\Gamma} + \gamma \left[\mathbf{P} + \mathbf{E}_n\right]_-$;
    Let $\mathbf{A} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\nabla g(\mathbf{P}, \mathbf{\Gamma})(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$;
    Let $\mathbf{v}$ be the smallest algebraic eigenvector of $\mathbf{A}$, such that $\mathbf{v}^\top \mathbf{1} = 0$;
    $\mathbf{H} \leftarrow (K-1)\mathbf{v}\mathbf{v}^\top$;
    $\alpha \leftarrow 2/(t + t_{\text{inner}} + 2)$;
    $\mathbf{P} \leftarrow (1-\alpha)\mathbf{P} + \alpha\mathbf{H}$;
  $\mathbf{\Gamma} \leftarrow \left[\mathbf{\Gamma} + \tau\left(\mathbf{P} + \mathbf{E}_n\right)\right]_-$;
  if *converged* **then** break ;
$\mathbf{Q} \leftarrow \mathbf{P} + \mathbf{E}_n$

- There is no need to need for a highly accurate eigenvector computation (Hazan, 2008). We use the Lanczos algorithm and set its accuracy to $(t+1)^{-1}$.
- Alg. 1 solves a maximization problem and requires the eigenvector with the largest algebraic eigenvalue. To solve the minimization problem (15a), we simply compute the eigenvector with the smallest algebraic eigenvalue (Jaggi, 2013).
- As $\mathbf{b} = \mathbf{1}$, we can enforce the orthogonality constraint ${\mathbf{v}_t}^\top \mathbf{1} = 0$ by computing the maximum eigenvalue of $\mathbf{A} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\nabla g(\mathbf{P}, \mathbf{\Gamma})(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$. This operation can be carried out very efficiently.

Complexity. The complexity of Alg. 1 is similar to that of Hazan's (2008), plus an additional factor to compute $\mathbf{A}$. From Proposition 1, Alg. 1 yields a solution with accuracy ε, i.e., $f(\mathbf{Z}_t) \geq f(\mathbf{Z}^\star) - \varepsilon$, in $\frac{4C_f}{\varepsilon} - 1$ iterations. Computing $\nabla g(\mathbf{P}, \mathbf{\Gamma})$, $\mathbf{A}$, and $\mathbf{H}_t$ require n^2 operations. Let T_{EIG} be the number of iterations of the eigensolver, each iteration taking $O(n^2)$ operations. Additional operations require $O(n)$ time. Then, the overall complexity of Alg. 1 is

$$O\left(\frac{C_f}{\varepsilon}\left[n + n^2 + n^2 T_{\text{EIG}}\right]\right). \tag{22}$$

For the Lanczos algorithm, and our accuracy setting of $(t+1)^{-1}$, we have $T_{\text{EIG}} = O((t+1)\log n)$. In this case, the complexity per iteration is $O(n^2 \log n)$. As a comparison, standard SDP solvers have a complexity of $O(n^3)$ per iteration. These solvers also involve significant memory usage, while our algorithm has an optimal space complexity of $O(n^2)$.

5.4. Experimental analysis

Throughout the iterations of Alg. 2, $\mathbf{P} \in \mathcal{P}_{k-1}$, see Eq. (18). Thus, we only need to keep track of the constraint $\mathbf{Q} = \mathbf{P} + \mathbf{E}_n \geq \mathbf{0}$ and of the value of the objective $\operatorname{Tr}(\mathbf{DP})$.

We illustrate with two typical examples the empirical convergence of these values in Fig. 13. the convergence the objective value is clearly superlinear, while we observe a linear convergence for the nonnegativity constraint. Accelerating the latter rate is an interesting line of future research.

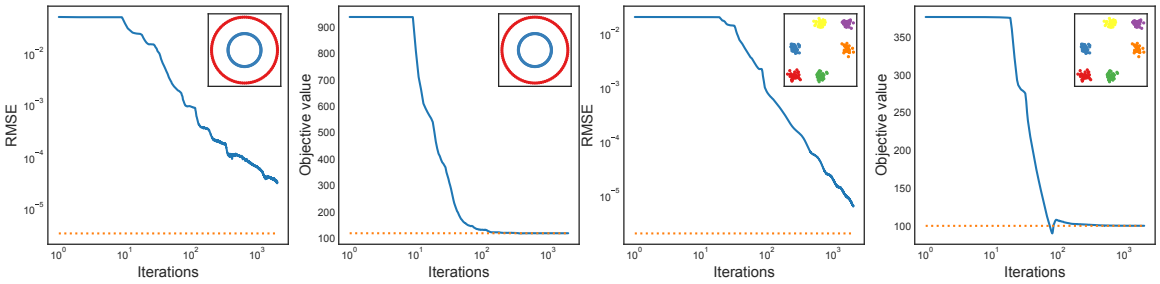


Figure 13: Prototypical examples of the behavior of the proposed conditional gradient NOMAD solver as its iterations progress. On the left plots, we show the RMSE of $[\mathbf{Q}]_- = [\mathbf{P} + \mathbf{E}_n]_-$, with the average computed over its non-zero entries. After about 10 iterations, the RMSE drops linearly, as usual for the method of multipliers. On the right plots, we display the objective value $\mathrm{Tr}\,(\mathbf{DP})$, which usually converges in a few hundred iterations. In each case, as a reference, we show in orange the values returned by the standard SDP solver. The proposed algorithm enforces the nonnegativity constraint in NOMAD less accurately (although accurate enough for practical purposes), while exactly enforcing all the other constraints.

We can see in Fig. 13 that standard solvers enforce the nonnegativity constraint more accurately. However, they do not exactly enforce $\mathbf{P} \in \mathcal{P}_{k-1}$. There is a trade-off between what can be enforced up to which precision, making the solutions sometimes not exactly comparable.

We show the suitability of the proposed NOMAD solver in Fig. 14. In the vast majority of cases the solutions are the same. While the proposed method enforces the nonnegativity constraint less accurately than the standard solver, it enforces all the other constraints exactly. This is why in the teapot example, bottom left of Fig. 14, the solution of the proposed method looks less jagged than the one of the standard solver: the constraint $\mathbf{Q1} = \mathbf{1}$ is more accurately enforced, resulting in a more "circulant" representation.

In Fig. 15, we present the speed comparison of computing NOMAD with three different methods: two state-of-the-art SDP solvers, SCS (O'Donoghue et al., 2016b) and SDPNAL+ (Yang et al., 2015), the low-rank Burer-Monteiro solver (discussed in Sec. 4), and the proposed conditional gradient method. The Burer-Monteiro method is the fastest. Keep in mind that the latter does not guarantee convergence to the global optimal solution; this is particularly true specially in its fastest setting, i.e., by keeping r relatively small, see Sec. 4. Among solvers that solve a convex problem, for very small problems (up to 250 points), standard SDP solvers are the fastest. For larger problems the proposed solver is significantly faster. It is important to point out that, in theory, the speed difference grows significantly larger. This is hard to show in practice as standard solvers either run out of memory very quickly (SCS) or are implemented to time out for big instances (SDPNAL+); the proposed solver has a much more efficient use of memory.

We highlight the extended computational capabilities of the proposed conditional gradient method with an example that cannot be handled by standard SDP solvers. We use as input the 9603×9603 Gramian formed by all (vectorized) images of the digit zero in MNIST. The proposed algorithm is able to compute a solution to NOMAD with ease for a problem size about 100 times larger than the upper size limit for standard solvers. In the 2D embedding of the solution (see Sec. 3 for details about its computation), shown in Fig. 16, we can clearly see that the images are organized by their intrinsic characteristics (elongation and rotation).

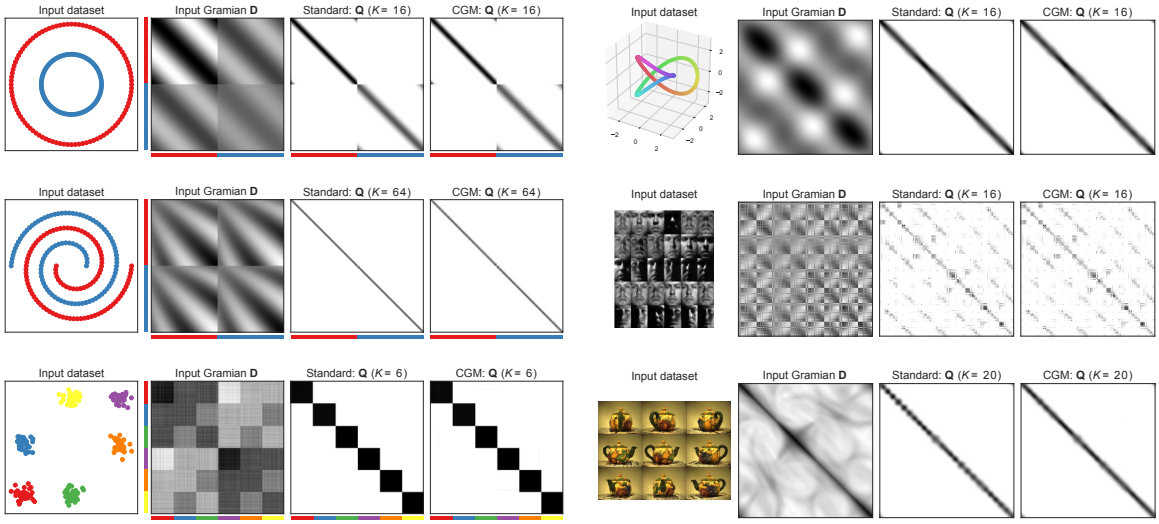


Figure 14: Comparison of the standard SDP solver with the proposed conditional gradient solver (CGM) for NOMAD on different datasets. In most cases, the results are practically indistinguishable while being delivered much faster.

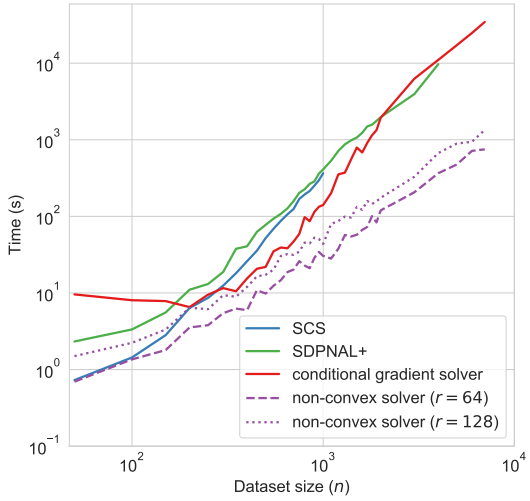


Figure 15: Running time comparison (smaller is better) of different NOMAD solvers for $K = 16$ (SCS (O'Donoghue et al., 2016b) and SDPNAL+ (Yang et al., 2015) are written in a highly optimized C/C++ code, while we use our non-optimized Python code for the others). The non-convex solver is much faster than the convex ones. Unfortunately, it may yield different results, see Fig. 12, and may not converge to the global maximum. The conditional gradient algorithm proposed in this paper is much faster than SCS and SDPNAL+ (about three times faster for $n = 10^3$) but guarantees converging to the global optimum. Additionally, the proposed algorithm handles large problems seamlessly: in our desktop with 128GB of RAM, SCS (running under CVXPY) runs out of memory with instances larger than $n = 1200$) while SDPNAL+ times out before converging for instances larger than $n = 4000$.

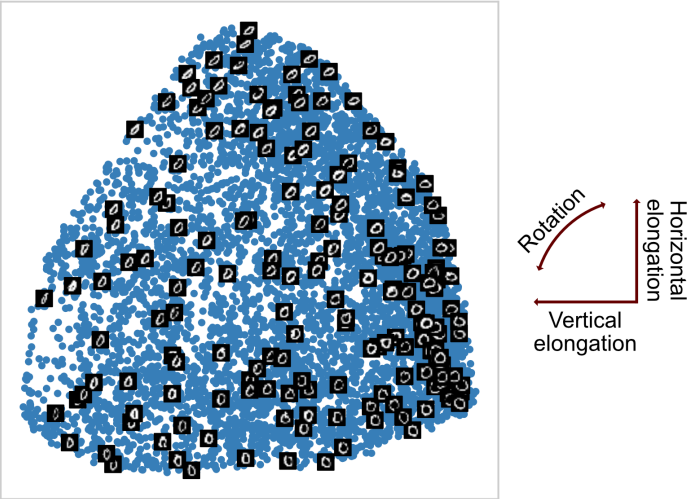


Figure 16: We show the 2D embedding of the digit 0 in MNIST, computed in the same fashion as in Fig. 6. In this case, we use all 9603 images of the digit and obtain a 9603×9603 matrix. We compute the solution of NOMAD with $K = 128$ using the proposed conditional gradient method (Alg. 2). In contrast, traditional SDP-solvers can only handle dense matrices approximately 100 times smaller. As in Fig. 6, the data gets organized according to different visual characteristics of the hand-written digit (e.g., orientation and elongation).

6. Conclusions

In this work, we showed that NOMAD can learn multiple low-dimensional data manifolds in high-dimensional spaces. An SDP instance, it is convex and can be solved in polynomial-time. Unlike most manifold learning algorithms, the user does not need to select/use a kernel and no nearest-neighbors searches are involved.

We also studied the computational performance of NOMAD. We first focused on a non-convex Burer-Monteiro-style algorithm and performed both theoretical and empirical analysis. Finally, we presented a new algorithm for NOMAD based on the conditional gradient method. The proposed algorithm is convex and yet efficient. This algorithm allows us, for the first time, to analyze the behavior of NOMAD on large datasets.

Related and future work. It has not escaped our attention that NOMAD can be considered as an instance of kernel alignment (Cristianini et al., 2002). In supervised setting, kernel alignment has been previously formulated as an SDP (e.g., Lanckriet et al., 2004; Cortes et al., 2012). Even beyond the distinction between the supervised and unsupervised scenarios, this body of work differs significantly from NOMAD. Its goal is to optimally combine pre-computed kernel matrices, whereas NOMAD learns such a matrix from scratch. Nonetheless, we find this connection with kernel learning very promising and plan to investigate it further in the future.

Acknowledgments

We thank Afonso Bandeira, Alexander Genkin, Victor Minden, and Cengiz Pehlevan for helpful discussions.

References

A. Amini and E. Levina. On semidefinite relaxations for the block model. Technical report, arXiv:1406.5647, 2014.

P. Awasthi, A. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. Relax, no need to round: Integrality of clustering formulations. In *ITCS*, 2015.

C. Bachoc, D. C. Gijswijt, A. Schrijver, and F. Vallentin. Invariant semidefinite programs. In *Handbook on semidefinite, conic and polynomial optimization*, pages 219–269. Springer, 2012.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

N. Boumal, V. Voroninski, and A. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. In *NIPS*, 2016.

R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Y. Cho and L. Saul. Kernel methods for deep learning. *NIPS*, pages 342–350, 2009.

K. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010.

C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel-target alignment. *NIPS*, 2002.

C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

E. Hazan. Sparse approximate solutions to semidefinite programs. In *LATIN*, 2008.

Y. Hou, J. Whang, D. Gleich, and I. Dhillon. Non-exhaustive, overlapping clustering via low-rank semidefinite programming categories and subject descriptors. In *KDD*, 2015.

T. Iguchi, D. Mixon, J. Peterson, and S. Villar. On the tightness of an SDP relaxation of k-means. Technical report, arXiv:1505.04778, 2015.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *ICML*, 2013.

A. Javanmard, A. Montanari, and F. Ricci-Tersenghi. Phase transitions in semidefinite relaxations. Technical report, arXiv:1511.08769, 2015.

M. Kaykobad. On nonnegative factorization of matrices. *Linear Algebra and its Applications*, 96: 27–33, 1987.

B. Kulis, A. Surendran, and J. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *AISTATS*, 2007.

G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5(Jan):27–72, 2004.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

J. Maxfield and H. Minc. On the matrix equation X' X= A. *Proceedings of the Edinburgh Mathematical Society (Series 2)*, 13(02):125–129, 1962.

D. Mixon, S. Villar, and R. Ward. Clustering subgaussian mixtures by semidefinite programming. Technical report, arXiv:1602.06612, 2016.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3): 1042–1068, 2016a.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3): 1042–1068, 2016b.

C. Pehlevan, A. Genkin, and D. Chklovskii. A clustering neural network model of insect olfaction. In *Asilomar*, 2017.

J. Peng and Y. Wei. Approximating K-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18(1):186–205, jan 2007.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2000.

W. So and C. Xu. When does CP-rank equal rank? Technical report, arXiv:1308.3193, 2013.

J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2000.

M. Tepper and G. Sapiro. A bi-clustering framework for consensus problems. *SIAM Journal on Imaging Sciences*, 7(4):2488–2525, 2014.

M. Tepper and G. Sapiro. Compressed nonnegative matrix factorization is fast and accurate. *IEEE Transactions on Signal Processing*, 64(9):2269–2283, 2016.

K. Weinberger and L. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. *AAAI*, 2006.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *NIPS*, 2008.

Y. Xu, W. Yin, Z. Wen, and Y. Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Front. Math. China*, 7(2):365–384, 2012.

L. Yang, D. Sun, and K. C. Toh. SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

Y. Yu, J. Neufeld, R. Kiros, X. Zhang, and D. Schuurmans. Regularizers versus losses for nonlinear dimensionality reduction. In *ICML*, 2012.

P. Zhang, Y. Ren, and B. Zhang. A new embedding quality assessment method for manifold learning. *Neurocomputing*, 97:251–266, 2012.

Appendix A. Relationship with K-means

K-means seeks to cluster a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ by computing

$$\min_{\boldsymbol{\mathcal{C}}_K} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \left\| \mathbf{x}_i - \tfrac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_j \in \mathcal{C}_k} \mathbf{x}_j \right\|_2^2, \tag{K-means}$$

where $\boldsymbol{\mathcal{C}}_K = \{\mathcal{C}_k\}_{k=1}^K$ is a partition of $\mathcal{X}$, i.e., $\mathcal{C}_k \cap \mathcal{C}_{k'} = \emptyset$ and $\bigcup_k \mathcal{C}_k = \mathcal{X}$. Albeit its popularity, it is known to be NP-Hard and, in practice, users employ an heuristic (Lloyd, 1982, originally developed in 1957) to find a solution. The objective function of K-means, henceforth denoted J_K, can be rewritten (dropping the terms that are constant with respect to $\mathcal{C}_k$) as

$$J_K = -\sum_{k=1}^{K} \tfrac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}_k} \mathbf{x}_i^\top \mathbf{x}_j. \tag{23}$$

Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the matrix formed by horizontally concatenating the vectors in $\mathcal{X}$. Let $\mathbf{z}_k \in \{0,1\}^n$ be the indicator vector of set $\mathcal{C}_k$. Let $\mathbf{Y}$ be the $k \times n$ matrix with rows $\mathbf{Y}_{k:} = |\mathcal{C}_k|^{-1/2} \mathbf{z}_k$. We have

$$\begin{aligned} J_K &= -\sum_{k=1}^{K} \tfrac{1}{|\mathcal{C}_k|} \sum_{i,j} \mathbf{x}_i^\top \mathbf{x}_j \cdot (\mathbf{z}_k^\top \mathbf{z}_k)_{ij} && \text{(24a)} \\ &= -\sum_{i,j} (\mathbf{X}^\top \mathbf{X})_{ij} (\mathbf{Y}^\top \mathbf{Y})_{ij} && \text{(24b)} \\ &= -\operatorname{Tr}\left(\mathbf{X}^\top \mathbf{X} \mathbf{Y}^\top \mathbf{Y}\right). && \text{(24c)} \end{aligned}$$

By construction, the matrix $\mathbf{Q} = \mathbf{Y}^\top \mathbf{Y}$ exhibits the following properties

$$\mathbf{Q}\mathbf{1} = \mathbf{1}, \tag{25}$$

$$\operatorname{Tr}(\mathbf{Q}) = K. \tag{26}$$

Let $\mathbf{D}$ be the Gramian matrix, i.e., $\mathbf{D} = \mathbf{X}^\top \mathbf{X}$. We can then re-cast K-means as the optimization problem

$$\max_{\mathbf{Y} \in \mathcal{V}_{\mathbf{Y}}^{k \times n}} \operatorname{Tr}(\mathbf{DQ}) \quad \text{s.t.} \quad \begin{array}{c} \mathbf{Q1} = \mathbf{1}, \\ \operatorname{Tr}(\mathbf{Q}) = K, \\ \mathbf{Q} = \mathbf{Y}^\top \mathbf{Y}. \end{array} \tag{27}$$

where $\mathcal{V}_{\mathbf{Y}} = \{0\} \cup \left\{ |\mathcal{C}_k|^{-1/2} \right\}_{k=1}^{K}$. Seeking to apply the desirable properties of SDP to K-means, we can pose (Kulis et al., 2007; Peng and Wei, 2007)

$$\max_{\mathbf{Q} \in \mathbb{R}^{n \times n}} \operatorname{Tr}(\mathbf{DQ}) \quad \text{s.t.} \quad \begin{array}{c} \mathbf{Q1} = \mathbf{1}, \\ \operatorname{Tr}(\mathbf{Q}) = K, \\ \operatorname{rank}(\mathbf{Q}) = K, \\ \mathbf{Q} \succeq 0,\ \mathbf{Q} \geq \mathbf{0}. \end{array} \tag{28}$$

where mixed-integer program is relaxed into the real-valued nonnegative program, directly optimizing over $\mathbf{Q}$. NOMAD is as a relaxation of this problem, simply obtained by removing the rank constraint.

Appendix B. On the complete positivity of NOMAD solutions on circulant matrices

Proposition 2 *If the solution $\mathbf{Q}_*$ to NOMAD is a circulant matrix, then it is CP for every $K \leq 3/2$ or $K \geq \frac{n}{2}$.*

Proof For $K \leq 3/2$, plugging the constraint $\mathbf{Q}_* \mathbf{1} = \mathbf{1}$ into Corollary 2.6 in (So and Xu, 2013, p. 7) gives the desired result.

Let us address $K \geq \frac{n}{2}$. Kaykobad (1987) proved that every diagonally dominant matrix $\mathbf{A}$, i.e., $|(\mathbf{A})_{ii}| \geq \sum_{j \neq i} |(\mathbf{A})_{ij}|$ for all i, is a CP matrix. We have to prove then that $\mathbf{Q}_* \geq \mathbf{0}$ is diagonally dominant. We have $\operatorname{Tr}(\mathbf{Q}_*) = K$ and, since $\mathbf{Q}_*$ is circulant, all $(\mathbf{Q}_*)_{ii}$ have the same value. Then, $(\mathbf{Q}_*)_{ii} = K/n$. From $\mathbf{Q}_* \mathbf{1} = \mathbf{1}$, $\sum_{j \neq i} (\mathbf{Q}_*)_{ij} = 1 - (\mathbf{Q}_*)_{ii} = 1 - K/n$. Hence, $\mathbf{Q}_*$ is diagonally dominant for $K \geq \frac{n}{2}$. ■

Appendix C. Additional results

We include additional results of the multi-layer NOMAD algorithm using different values of k in each layer.

Appendix D. Symmetric NMF

In this section, we present the algorithm used to compute the symmetric NMF of a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, defined as

$$\min_{\mathbf{Y} \in \mathbb{R}^{n \times r}} \left\| \mathbf{A} - \mathbf{Y}\mathbf{Y}^\top \right\|_F^2 \quad \text{s.t.} \quad \mathbf{Y} \geq \mathbf{0}. \tag{SNMF}$$

We use the alternating direction method of multipliers (ADMM) to solve it. In short, ADMM solves convex optimization problems by breaking them into smaller subproblems, which are individually

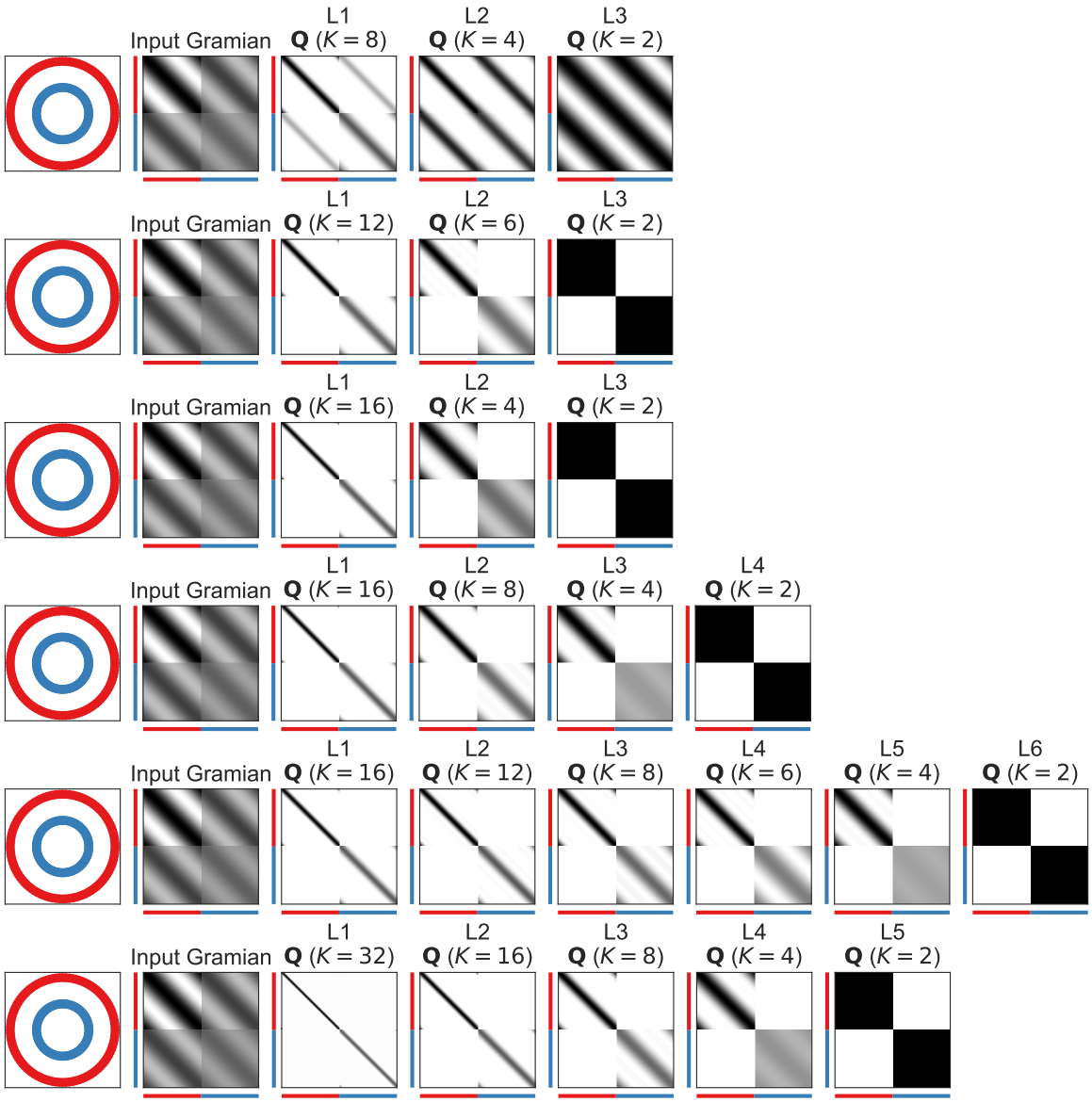


Figure 17: Additional results of multi-layer NOMAD.

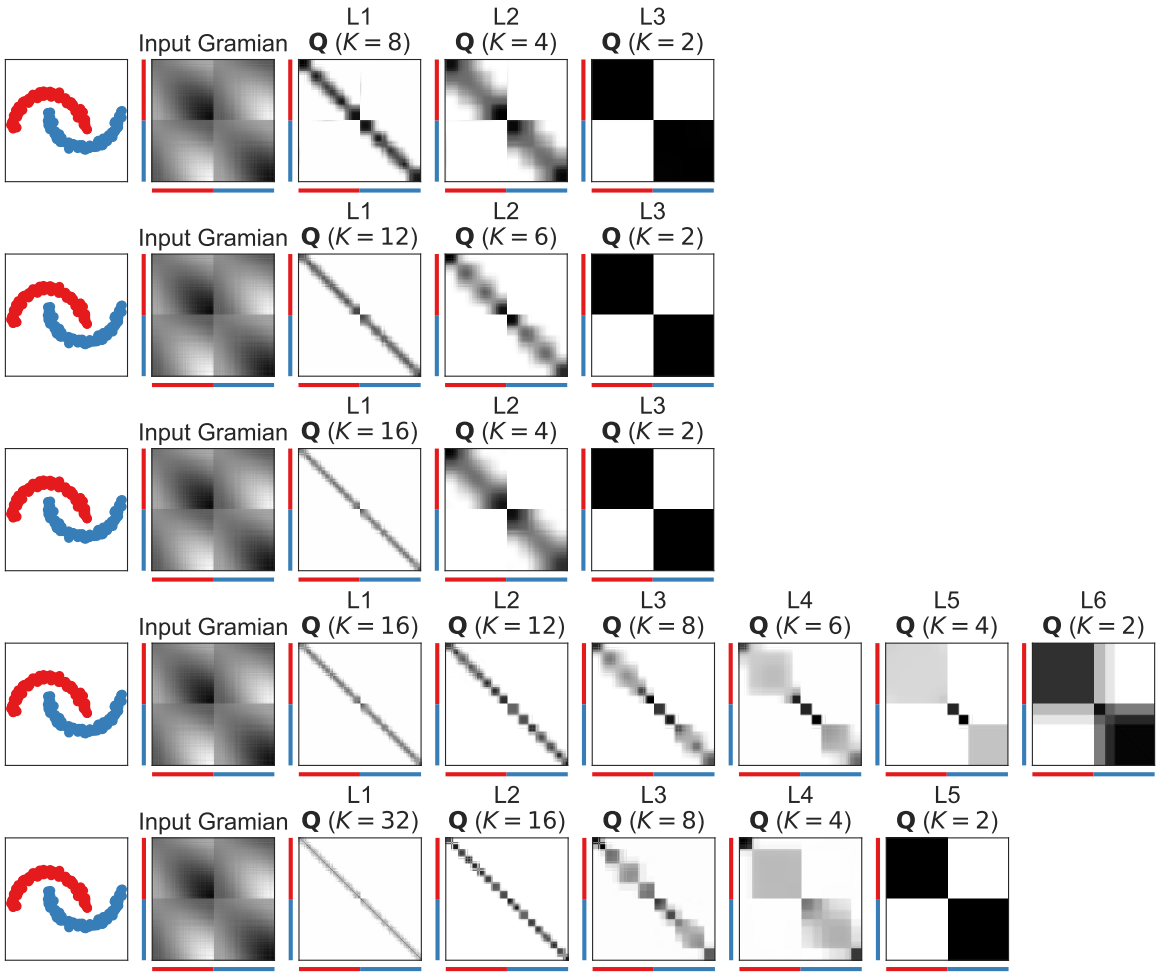


Figure 18: Additional results of multi-layer NOMAD.

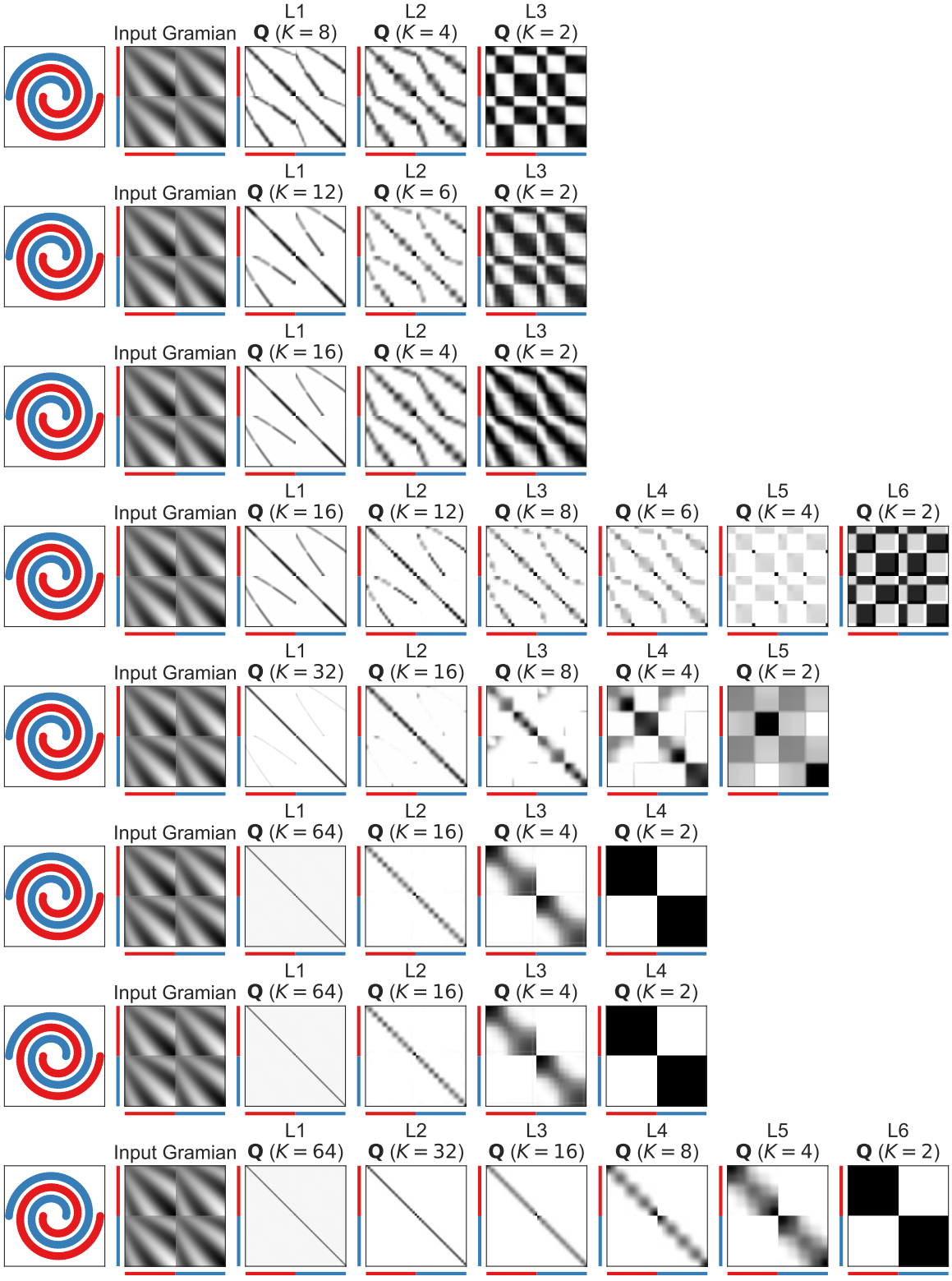


Figure 19: Additional results of multi-layer NOMAD.

easier to handle. It has also been extended to handle non-convex problems, e.g., to solve several flavors of NMF (Févotte and Idier, 2011; Xu et al., 2012; Tepper and Sapiro, 2014, 2016).

Problem (SNMF) can be equivalently re-formulated as

$$\min_{\mathbf{Y}\in\mathbb{R}^{n\times r}} \left\|\mathbf{A}-\mathbf{Y}\mathbf{X}^\top\right\|_F^2 \quad \text{s.t.} \quad \mathbf{Y}=\mathbf{X},\ \mathbf{Y}\geq\mathbf{0},\ \mathbf{X}\geq\mathbf{0}, \tag{29}$$

and we consider its augmented Lagrangian,

$$\mathscr{L}\left(\mathbf{X},\mathbf{Y},\mathbf{\Gamma}\right)=\tfrac{1}{2}\left\|\mathbf{A}-\mathbf{Y}\mathbf{X}^\top\right\|_F^2+\tfrac{\sigma}{2}\left\|\mathbf{Y}-\mathbf{X}\right\|_F^2-\operatorname{Tr}\left(\mathbf{\Gamma}^\top\left(\mathbf{Y}-\mathbf{X}\right)\right) \tag{30}$$

where $\mathbf{\Gamma}$ is a Lagrange multiplier, σ is a penalty parameter.

The ADMM algorithm works in a coordinate descent fashion, successively minimizing $\mathscr{L}$ with respect to $\mathbf{X}, \mathbf{Y}$, one at a time while fixing the other at its most recent value and then updating the multiplier $\mathbf{\Gamma}$. For the problem at hand, these steps are

$$\mathbf{X}^{(t+1)}=\operatorname*{argmin}_{\mathbf{X}\geq\mathbf{0}}\mathcal{L}\left(\mathbf{X},\mathbf{X}^{(t)},\mathbf{\Gamma}^{(t)}\right), \tag{31a}$$

$$\mathbf{Y}^{(t+1)}=\operatorname*{argmin}_{\mathbf{Y}\geq\mathbf{0}}\mathcal{L}\left(\mathbf{X}^{(t+1)},\mathbf{Y},\mathbf{\Gamma}^{(t)}\right), \tag{31b}$$

$$\mathbf{\Gamma}^{(t+1)}=\mathbf{\Gamma}^{(t)}-\eta\sigma\left(\mathbf{X}^{(t+1)}-\mathbf{Y}^{(t+1)}\right). \tag{31c}$$

In our experiments, we fix η and σ to 1. We initialize the algorithm with a random matrix.

Appendix E. Non-convex SDP solver

We follow the algorithm proposed in Kulis et al. (2007); Hou et al. (2015) to solve Problem (11). Our approach has a small but fundamental difference: instead of setting $r=K$, we allow for $r\geq K$. We define the augmented Lagrangian of Problem (11) as

$$\begin{aligned}\mathcal{L}\left(\mathbf{Y},\mu,\lambda\right)=&-\operatorname{Tr}\left(\mathbf{D}\mathbf{Y}^\top\mathbf{Y}\right)+\tfrac{\varphi}{2}\left\|\mathbf{Y}^\top\mathbf{Y}\mathbf{1}-\mathbf{1}\right\|_2^2-\mu^\top(\mathbf{Y}^\top\mathbf{Y}\mathbf{1}-\mathbf{1})\\&+\tfrac{\varphi}{2}(\operatorname{Tr}\left(\mathbf{Y}^\top\mathbf{Y}\right)-K)^2-\lambda(\operatorname{Tr}\left(\mathbf{Y}^\top\mathbf{Y}\right)-K),\end{aligned} \tag{32}$$

where μ, λ are Lagrange multipliers, σ, φ are penalty parameters. We obtain $\mathbf{Y}$ by running the steps

$$\mathbf{Y}^{(t+1)}=\operatorname*{argmin}_{\mathbf{Y}\geq\mathbf{0}}\mathcal{L}\left(\mathbf{Y},\mu^{(t)},\lambda^{(t)}\right), \tag{33a}$$

$$\mu^{(t+1)}=\mu^{(t)}-\eta\sigma\left(\mathbf{Y}^\top\mathbf{Y}\mathbf{1}-\mathbf{1}\right), \tag{33b}$$

$$\lambda^{(t+1)}=\lambda^{(t)}-\eta\varphi\left(\operatorname{Tr}\left(\mathbf{Y}^\top\mathbf{Y}\right)-K\right). \tag{33c}$$

This is a non-standard approach since the minimization over $\mathbf{Y}$ (the gradient $\partial\mathcal{L}/\partial\mathbf{Y}$ is given in Hou et al. (2015)) is a non-convex problem. Although there are no guarantees about the convergence of the procedure, theoretical assurances for related problems have been presented in Boumal et al. (2016). To perform the minimization with respect to $\mathbf{Y}$, we use the L-BFGS-B algorithm (Byrd et al., 1995) with bound constraints ($(\mathbf{Y})_{ij}\in[0,1]$). Finally, the initialization to the overall iterative algorithm is done with symmetric nonnegative matrix factorization, see Appendix D. In our experiments, we fix η, φ, and σ to 1 and prenormalize $\mathbf{D}$ (dividing by its Frobenius norm).

Appendix F. Proofs for the conditional gradient algorithm

This section closely follows the works of Hazan (2008) and Clarkson (2010).

Lemma 3 *The dual objective of*

$$\max_{\mathbf{Z}} f(\mathbf{Z}) \quad \textit{s.t.} \quad \mathbf{Z}\mathbf{b} = \mathbf{0}, \quad \operatorname{Tr}(\mathbf{Z}) = k-1, \quad \mathbf{Z} \succeq \mathbf{0} \tag{34}$$

is

$$\min_{\mathbf{Z}} w(\mathbf{Z}), \tag{35}$$

where

$$w(\mathbf{Z}) = s\,\phi(\mathbf{Z}) + f(\mathbf{Z}) - \operatorname{Tr}(\mathbf{Z}\nabla f(\mathbf{Z})), \tag{36}$$

$$\phi(\mathbf{Z}) = \max_{\substack{\|\mathbf{v}\|_2=1 \\ \mathbf{v}^\top\mathbf{b}=0}} \mathbf{v}^\top \nabla f(\mathbf{Z})\mathbf{v}. \tag{37}$$

Proof The Lagrangian relaxation of Problem (19) is

$$\begin{aligned} \ell = - \max_{\boldsymbol{\Psi},\psi,\mathbf{y}} \min_{\mathbf{Z}} \; & - f(\mathbf{Z}) - \operatorname{Tr}(\boldsymbol{\Psi}\mathbf{Z}) \\ & + \psi(\operatorname{Tr}(\mathbf{Z}) - s) + \mathbf{y}^\top \mathbf{Z}\mathbf{b}, \end{aligned} \tag{38}$$

where $\boldsymbol{\Psi} \succeq \mathbf{0}$, $\psi \in \mathbb{R}$, and $\mathbf{y} \in \mathbb{R}^n$ are Lagrange multipliers. Differentiating and equating to zero we get $\boldsymbol{\Psi} = -\nabla f(\mathbf{Z}) + \psi\mathbf{I} + \mathbf{y}\mathbf{b}^\top$. Note that $\boldsymbol{\Psi} \succeq \mathbf{0}$ implies that $\mathbf{y} = c\mathbf{b}$ for some $c \in \mathbb{R}$, yielding

$$\boldsymbol{\Psi} = -\nabla f(\mathbf{Z}) + \psi\mathbf{I} + c\mathbf{b}\mathbf{b}^\top \succeq \mathbf{0}. \tag{39}$$

Plugging Eq. (39) and $\mathbf{y} = c\mathbf{b}$ in Eq. (38) we get

$$\begin{aligned} \ell &= \max_{\psi} \min_{\mathbf{Z}} f(\mathbf{Z}) - \operatorname{Tr}\left(\left(\nabla f(\mathbf{Z}) + \psi\mathbf{I} + c\mathbf{b}\mathbf{b}^\top\right)\mathbf{Z}\right) \\ &\qquad + \psi(\operatorname{Tr}(\mathbf{Z}) - s) + c\mathbf{b}^\top\mathbf{Z}\mathbf{b} && (40) \\ &= \max_{\psi\in\mathbb{R}} \min_{\mathbf{Z}} f(\mathbf{Z}) - \operatorname{Tr}(\nabla f(\mathbf{Z})\,\mathbf{Z}) + \psi s && (41) \\ &= \max_{\substack{\mathbf{Z} \\ \psi\in\mathbb{R}}} f(\mathbf{Z}) - \operatorname{Tr}(\nabla f(\mathbf{Z})\,\mathbf{Z}) - \psi s. && (42) \end{aligned}$$

From Eq. (39), $\psi\mathbf{I} \succeq \nabla f(\mathbf{Z}) - c\mathbf{b}\mathbf{b}^\top$, implying

$$\begin{aligned} \psi &\geq \lambda_{\max}\left\{\nabla f(\mathbf{Z}) - c\mathbf{b}\mathbf{b}^\top\right\} && (43a) \\ &\geq \lambda_{\max}\left\{\nabla f(\mathbf{Z}) - d\mathbf{b}\mathbf{b}^\top\right\} \quad \forall d > c && (43b) \\ &\geq \phi(\mathbf{Z}). && (43c) \end{aligned}$$

The last inequality comes from taking $d \to +\infty$, thus shifting the eigenvalue associated with $\mathbf{b}$ (should there be one) away from the maximum. Without loss of generality, we set $\psi = \phi(\mathbf{Z})$, finally obtaining $\ell = \min_{\mathbf{Z}} w(\mathbf{Z})$.

Proposition 4 *Let* $\mathbf{X}, \mathbf{Z} \in \mathcal{P}_s$ *and* $\mathbf{Y} = \mathbf{X} + \alpha(\mathbf{Z} - \mathbf{X})$ *and* $\alpha \in \mathbb{R}$*. The curvature constant of* f *is*

$$C_f := \sup_{\mathbf{X},\mathbf{Z},\alpha} \tfrac{1}{\alpha^2}[f(\mathbf{X}) - f(\mathbf{Y}) + (\mathbf{Y} - \mathbf{X}) \bullet \nabla f(\mathbf{X})]. \tag{44}$$

Let $\mathbf{Z}^\star$ *be the solution to*

$$\max_{\mathbf{Z}} f(\mathbf{Z}) \quad \textit{s.t.} \quad \mathbf{Z}\mathbf{b} = \mathbf{0}, \quad \mathrm{Tr}\,(\mathbf{Z}) = k - 1, \quad \mathbf{Z} \succeq \mathbf{0}. \tag{45}$$

The iterates $\mathbf{Z}_t$ *of Alg. 1 satisfy for all* $t > 1$

$$f(\mathbf{Z}^\star) - f(\mathbf{Z}_t) \leq \tfrac{8C_f}{t+2}. \tag{46}$$

Proof Let

$$h(\mathbf{Z}) = \tfrac{1}{4C_f}[f(\mathbf{Z}^\star) - f(\mathbf{Z})]. \tag{47}$$

Proving that $h(\mathbf{Z}_t) \leq \frac{2}{t+2}$ yields the desired result. From Lemma 3, we have

$$w(\mathbf{Z}) \geq w(\mathbf{Z}^\star) \geq f(\mathbf{Z}^\star) \geq f(\mathbf{Z}). \tag{48}$$

By eqs. (36) and (37), we have

$$\mathbf{v}_t^\top \nabla f(\mathbf{Z}_t)\mathbf{v}_t = \phi(\mathbf{Z}) \tag{49}$$

$$= w(\mathbf{Z}_t) - f(\mathbf{Z}_t) + \mathrm{Tr}\,(\mathbf{Z}\,\nabla f(\mathbf{Z}_t))\,. \tag{50}$$

Therefore,

$$\mathrm{Tr}\,((\mathbf{H}_t - \mathbf{Z}_t)\,\nabla f(\mathbf{Z}_t)) = w(\mathbf{Z}_t) - f(\mathbf{Z}_t). \tag{51}$$

Now, using Eq. (20)

$$f(\mathbf{Z}_{t+1}) = f(\mathbf{Z}_t + \alpha_t(\mathbf{H}_t - \mathbf{Z}_t)) \tag{52a}$$

$$\geq f(\mathbf{Z}_t) + \alpha_t\,\mathrm{Tr}\,((\mathbf{H}_t - \mathbf{Z}_t)\,\nabla f(\mathbf{Z}_t)) + \alpha^2 C_f. \tag{52b}$$

Putting Eq. (52b) together with eqs. (48) and (51),

$$f(\mathbf{Z}_{t+1}) \geq f(\mathbf{Z}_t) + \alpha_t(w(\mathbf{Z}_t) - f(\mathbf{Z}_t)) - \alpha_t^2 C_f \tag{53a}$$

$$\geq f(\mathbf{Z}_t) + \alpha_t(f(\mathbf{Z}^\star) - f(\mathbf{Z}_t)) - \alpha_t^2 C_f \tag{53b}$$

$$= f(\mathbf{Z}_t) + 4\alpha_t C_f h(\mathbf{Z}_t) - \alpha_t^2 C_f. \tag{53c}$$

From the definition of $h(\mathbf{Z}_t)$, this implies

$$h(\mathbf{Z}_{t+1}) \leq h(\mathbf{Z}_t) - \alpha_t h(\mathbf{Z}_t) + \tfrac{\alpha_t^2}{4}. \tag{54}$$

Finally, we prove inductively that $h(\mathbf{Z}_t) \leq \frac{2}{t+2}$. In the first iteration, $\alpha_1 = 1$ and $h(\mathbf{Z}_2) \leq \frac{1}{4}$. By taking $\alpha = \frac{2}{t+2}$, we have

$$h(\mathbf{Z}_{t+1}) \leq h(\mathbf{Z}_t) - \alpha_t h(\mathbf{Z}_t) + \tfrac{\alpha_t^2}{4} \tag{55a}$$

$$\leq (1 - \alpha_t)h(\mathbf{Z}_t) + \tfrac{\alpha_t^2}{2} \tag{55b}$$

$$= (1 - \tfrac{2}{t+2})\tfrac{2}{t+2} + \tfrac{2}{(t+2)^2} \leq \tfrac{2}{t+3}. \tag{55c}$$