Abstract

We consider dimensionality reduction for data sets with two or more independent degrees of freedom. For example, measurements of deformable shapes with several parts that move independently fall under this characterization. Mathematically, if the space of each continuous independent motion is a manifold, then their combination forms a product manifold. In this paper, we present an algorithm for manifold factorization given a sample of points from the product manifold. Our algorithm is based on spectral graph methods for manifold learning and the separability of the Laplacian operator on product spaces. Recovering the factors of a manifold yields meaningful lower-dimensional representations, allowing one to focus on particular aspects of the data space while ignoring others. We demonstrate the potential use of our method for an important and challenging problem in structural biology: mapping the motions of proteins and other large molecules using cryo-electron microscopy data sets.

1 Introduction

Consider a data-generating process $T$ which maps vectors of latent (unobserved) variables to observations,

$$ (\theta_1, \ldots, \theta_m) \xrightarrow{T} x \in \mathbb{R}^D. \quad (1) $$

Our focus is on the case in which the latent variables are low-dimensional and the observations are high-dimensional vectors. As an illustrative example, consider an industrial articulated robot where the latent variables $\theta_1, \ldots, \theta_m$ correspond to the angles of its $m$ rotary joints, and the observed vector $x$ is a set of measurements recorded by an external monitoring system. When $T$ is deterministic (i.e. without noise), smooth and injective, then the space of observations is a submanifold of $\mathbb{R}^D$ (Lee, 2012, Theorem 4.14). This observation underlies the field of manifold learning (Tenenbaum et al., 2000; Belkin and Niyogi, 2003; Coifman and Lafon, 2006; Talmon et al., 2013). A key problem in this field is dimensionality reduction, where a set of high-dimensional observations $x_1, \ldots, x_n \in \mathbb{R}^D$ is mapped to a low-dimensional space, hopefully one whose dimension is not much larger than the dimension of the latent space.

In this work, we explore the idea of manifold learning when the latent space is a product manifold. If each latent variable $\theta_i$ lies on a manifold $\mathcal{M}_i$ of dimension $d_i$ then the latent space is the Cartesian product

$$ \mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_m, \quad (2) $$

which is a manifold of dimension $d = d_1 + \cdots + d_m$. We refer to the decomposition (2) as a manifold factorization of $\mathcal{M}$ and to each of the manifolds $\mathcal{M}_i$ as factors. In this paper, we present a data-dependent algorithm for manifold factorization based on spectral graph-based methods for manifold learning (Belkin and Niyogi, 2003; Coifman and Lafon, 2006). These methods output a set of graph Laplacian eigenvectors which approximate the eigenfunctions of the Laplacian differential operator on the manifold. By mapping the observations to their eigenvector coordinates one can obtain a low-dimensional embedding of the data space (Bates, 2014). On a product manifold, it is well-known that the Laplacian eigenfunctions are nothing but pointwise products of eigenfunctions on the factor manifolds, one eigenfunction from each factor (see Section 2). The eigenfunctions are typically not observable; however, they can be approximated by the eigenvectors of a data-dependent graph Laplacian. All that is left is to find an approximate factorization of the set of Laplacian eigenvectors. We describe our particular approach for this factorization in Section 3.

In traditional methods for dimensionality reduction, each data point is mapped to a point in a low-dimensional vector space. In contrast to this, after running our algorithm, each point is mapped to a low-dimensional product space, where each factor encodes...
a different aspect of the data. This opens the door to exciting new methods for data visualization and representation. One can, for instance, use it to visualize observations that come from a 5-dimensional latent space as pairs of points on a 3-dimensional and a 2-dimensional manifold, associate different interpretations to the different factors, and ignore factors that correspond to nuisance variables.

In this paper, we mainly focus on products of two manifolds. However, extensions to more than two manifolds are briefly discussed in Section 5.

1.1 Related Work

One work that is closely related to ours is the spectral method of Singer (2006a) for linear independent component analysis (ICA), as it is also based on the separability of the Laplacian on a product space. This was later extended to a method for non-linear ICA that assumes an explicit product manifold. However, recent manifold learning (Belkin et al., 2012). The following summarizes some known results from the spectral theory of Riemannian manifolds.

Theorem 1. Let \( M \) be a compact Riemannian manifold, either without boundary or with Neumann boundary conditions. The solutions to the Helmholtz equation (3) satisfy the following:

(i) The eigenvalues are real, non-negative, and tend to infinity, \( 0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \to \infty \).

(ii) There exists a complete orthonormal basis for \( L^2(M) \) of real eigenfunctions \( \{f_k\}_{k=1}^\infty \).

2.2 The Laplacian over product manifolds

We now describe the behavior of the Laplacian operator on a product space. We begin with an example.

Example 1 (2D rectangle). Consider the rectangular domain \([0,a] \times [0,b] \subset \mathbb{R}^2\). Its eigenfunctions, with Neumann boundary conditions, are

\[
f_{m,n}(x,y) = \cos\left(\frac{m\pi}{a} x\right) \cos\left(\frac{n\pi}{b} y\right),
\]

\[
m, n \in \{0, 1, 2, \ldots\}.
\]

The corresponding eigenvalues are

\[
\lambda_{m,n} = \lambda_m + \lambda_n = \pi^2 \left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right).
\]

The eigenfunctions are precisely the products of eigenfunctions on the closed intervals \([0,a]\) and \([0,b]\), which are \(\cos(\frac{m\pi}{a} x)\) and \(\cos(\frac{n\pi}{b} y)\) and the eigenvalues of the product space are sums of the corresponding eigenvalues on the intervals. See Figure 1.

This example generalizes to any product manifold.

Theorem 2. Let \( M_1, M_2 \) be compact Riemannian manifolds where the eigenfunctions of \( M_i \) are \( \{f^{(i)}_k\}_{k=1}^\infty \) and the corresponding eigenvalues are \( \{\lambda^{(i)}_k\}_{k=1}^\infty \). Let \( \pi^{(i)} : M \to M_i \) denote the canonical projection of \( M \) onto \( M_i \). Then the eigenfunctions of \( M = M_1 \times M_2 \) are given by the pointwise products

\[
f_{m,n} = (f^{(1)}_m \circ \pi^{(1)})(f^{(2)}_n \circ \pi^{(2)}).
\]

The corresponding eigenvalues are the sums

\[
\lambda_{m,n} = \lambda^{(1)}_m + \lambda^{(2)}_n.
\]
For a proof, see (Canzani, 2013, Section 4.6).

By repeated factorization, Theorem 2 extends to products of more than two manifolds. The eigenfunctions of \( M_1 \times \ldots \times M_m \) are

\[
f_{k_1, \ldots, k_m} = \prod_{i=1}^{m} \left( f^{(i)}_{k_i} \circ \pi^{(i)} \right),
\]

where \( k_i \) is an index to the eigenfunctions of \( M_i \). The corresponding eigenvalues are

\[
\lambda_{k_1, \ldots, k_m} = \sum_{i=1}^{m} \lambda^{(i)}_{k_i}.
\]

### 2.3 Graph Laplacians and diffusion maps

The results of the previous sections hold for the Laplacian operator on a manifold. However, in typical data analysis settings we know neither the geometry of the manifold nor its Laplacian, but only have a sample of observations \( x_1, \ldots, x_n \in M \subset \mathbb{R}^D \). To estimate the eigenfunctions, we turn to the discrete graph Laplacian. Consider the sample to be the vertex set of a weighted graph \( V = \{x_1, \ldots, x_n\} \) and let \( W \in \mathbb{R}^{n \times n} \) be the matrix of edge weights. Gaussian affinities are often used for the weights,

\[
W_{ij} = \exp \left( - \|x_i - x_j\|^2 / (2\sigma^2) \right).
\]

Let \( D \) be the diagonal degree matrix that satisfies \( D_{ii} = \sum_j W_{ij} \). The (unweighted) graph Laplacian is

\[
L := D - W.
\]

Several variants of the graph Laplacian were proposed, including the symmetric-normalized graph Laplacian and the random-walk Laplacian

\[
L_{\text{sym}} := D^{-1/2} L D^{-1/2} \tag{14}
\]

\[
L_{\text{rw}} := D^{-1} L = I - D^{-1} W. \tag{15}
\]

Both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) are positive semi-definite and have the same spectrum (Von Luxburg, 2007),

\[
0 = \nu_1 \leq \nu_2 \leq \ldots \leq \nu_n. \tag{16}
\]

Defining \( A = D^{-1/2} W D^{-1/2} \), the limit \( A^{t/\sigma} \) as \( \sigma \to 0 \) is the heat operator \( H_t = e^{-t\Delta_M} \) for all \( t \in \mathbb{R} \). Thus, the eigenvalues \( \lambda_i \) of the continuous Laplace operator \( \Delta_M \) and the eigenvalues \( \nu_i \) of \( L_{\text{rw}} \) and \( L_{\text{sym}} \) satisfy

\[
\lim_{\sigma \to 0} \nu_i^{t/\sigma} = e^{-t\lambda_i}.
\]

So to compute the spectrum of \( \Delta_M \), we simply log-transform \( \nu_2, \ldots, \nu_n \) (Lafon, 2004, Section 2.3).

Denote the eigenvectors (which can be chosen to be real) that correspond to \( \nu_1, \ldots, \nu_n \) by \( \phi_1, \ldots, \phi_n \in \mathbb{R}^n \). We view them as real functions on the set of input points \( \phi_i : \{x_1, \ldots, x_n\} \to \mathbb{R} \). Viewed in this way, Laplacian eigenvectors define an orthogonal Fourier-like basis of oscillating functions (Lee and Izbicki, 2016). Laplacian eigenvectors can be used for dimensionality reduction by the following map,

\[
x_i \mapsto [\phi_2(x_i), \phi_3(x_i), \ldots, \phi_{\ell}(x_i)].
\]

This approach was pioneered by Belkin and Niyogi (2003) under the name Laplacian eigenmaps and later analyzed and extended by Coifman and Lafon (2006) under the diffusion maps framework.

### 2.4 Asymptotics of graph Laplacians

A graph Laplacian constructed from an i.i.d. sample of points on a manifold converges to a linear differential operator. For affinities that use the Euclidean distance in combination with a decay kernel (as in Eq. (12)) and when the points are drawn uniformly, the limiting operator is the Laplace-Beltrami operator \( \Delta_M \) (Hein et al., 2005; Giné and Koltchinskii, 2006; Singer, 2006b; Belkin and Niyogi, 2008). In the more general case, when the data has a non-uniform density \( p(x) \) then the graph Laplacian converges to a Fokker-Planck operator that has an additional drift term given by the log-density \( U(x) = -2 \log p(x) \) (Nadler et al., 2005; Coifman and Lafon, 2006),

\[
\Delta_M f = -\nabla U \cdot \nabla f. \tag{18}
\]

In addition to the pointwise convergence, several works have proved the spectral convergence of graph Laplacians, i.e. the convergence of their eigenvalues and eigenvectors (von Luxburg et al., 2008; Rosasco et al., 2010; García Trillos and Slepčev, 2018; García Trillos et al., 2020). The convergence of the eigenvectors is the key to our approach.

Alternative graph constructions, such as the Euclidean \( k \)-nearest-neighbor graph also converge to a Fokker-Planck operator (Ting et al., 2010). If a non-Euclidean norm is used to define affinities then the graph Laplacian does not converge to a Fokker-Planck operator, but to some non-intrinsic second-order differential operator (Kileel et al., 2020).
3 Method

As discussed in Section 2.2, eigenfunctions \( \varphi_k \) on the product space \( M_1 \times M_2 \) are pointwise products of an eigenfunction \( \varphi_i \) on \( M_1 \) and an eigenfunction \( \varphi_j \) on \( M_2 \). Combining this fact with the convergence of graph Laplacians (discussed in Section 2.4), we conclude the following: if we construct a graph Laplacian from a large sample of points on \( M \), then each Laplacian eigenvector \( \varphi_k \) should be approximately equal (up to normalization) to an element-wise product \( \varphi_i \varphi_j \) where \( \varphi_i \) and \( \varphi_j \) are vectors that approximate eigenfunctions on \( M_1 \) and \( M_2 \) respectively. Recall that the multiplicity of the eigenvalue \( \lambda = 0 \) is the number of connected components. It follows that for manifolds with a single connected component, the only eigenfunctions with \( \lambda = 0 \) are the constant functions. If either \( \varphi_i \) or \( \varphi_j \) correspond to a constant eigenfunction with eigenvalue \( \lambda = 0 \) we call \( \varphi_k \) a factor eigenvector. Otherwise, we call \( \varphi_k \approx \varphi_i \varphi_j \) a product eigenvector.

The main idea behind our method is to first differentiate between the factor eigenvectors and the product eigenvectors and then to divide the factor eigenvectors into two sets that correspond to the eigenspaces of \( M_1 \) and \( M_2 \). The division scheme is based on the observation that if \( \varphi_k \approx \varphi_i \varphi_j \) then this suggests that \( \varphi_i \) and \( \varphi_j \) belong to different factor manifolds. Mathematically, our method is based on two assumptions:

**Assumption 1.** The mapping of the latent space to the observations \( \{ \theta_1, \ldots, \theta_m \} \mapsto x \in \mathbb{R}^D \) is an isometric embedding into \( \mathbb{R}^D \).

**Remark 1.** As isometric embeddings preserve the Riemannian metric, the spectral properties are not affected by the particular embedding.

**Assumption 2.** The latent variables \( \theta_1, \ldots, \theta_m \) are drawn independently of each other.

**Remark 2.** The last assumption may be relaxed by using the diffusion map normalization for the graph Laplacian (Coifman and Lafon, 2006). This graph Laplacian converges to the Laplace-Beltrami operator.

3.1 Factorizing product eigenvectors

Suppose we have the first \( N \) non-trivial eigenvectors \( \varphi_1, \varphi_2, \ldots, \varphi_N \) of \( L_{rw} \), sorted by the corresponding log-transformed eigenvalues \( 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \) which approximate the spectrum of \( \Delta \). For each eigenvector \( \varphi_k \), we find the pair \((\varphi_i, \varphi_j)\) whose element-wise product is closest. This “closeness” is measured by the absolute cosine similarity,

\[
S(\varphi_k, \varphi_i \varphi_j) := \frac{|\langle \varphi_k, \varphi_i \varphi_j \rangle|}{\|\varphi_k\| \|\varphi_i \varphi_j\|} \in [0,1]. \tag{19}
\]

We take the absolute value of the dot product to account for the sign ambiguity of eigenvectors. For each \( k \), we find the combination \((i,j)\) with the maximal similarity. To avoid computing the element-wise product for all \( k(k-1)/2 \) combinations, we skip triplets \((i,j,k)\) for which \( |\lambda_i + \lambda_j - \lambda_k| > \delta \) for some threshold \( \delta > 0 \). This *eigenvalue criterion* is based on the results of Theorem 2. We additionally apply an *eigenvector criterion* that filters out triplets whose similarity is less than \( \gamma \). This is summarized in Algorithm 1.

**Algorithm 1: Identification of individual factors**

**Data:** Non-trivial eigenvectors \( \{\varphi_1, \ldots, \varphi_N\} \) of \( L_{rw} \), sorted increasingly by their log-transformed eigenvalues \( \lambda_1, \ldots, \lambda_N \).

**Result:** List of triplets \((i,j,k)\) where \( \varphi_k \approx \varphi_i \varphi_j \) and the corresponding similarity scores.

```plaintext
for k ← 1...N do
    maxS ← 0;
    for i, j < k do
        if |\(\lambda_i + \lambda_j - \lambda_k\)| < \(\delta\) and
            \(|S(\varphi_k, \varphi_i \varphi_j)| > maxS\) then
            maxS ← \(S(\varphi_k, \varphi_i \varphi_j)\);
            \(i_{\text{max}} \leftarrow i;\)
            \(j_{\text{max}} \leftarrow j;\)
        end
    end
    if maxS > \(\gamma\) then
        add \((i_{\text{max}}, j_{\text{max}}, k)\) to triplets;
    end
end
```

The result of the algorithm is a list of index triplets \((i,j,k)\), which indicate that the best factorization we could find for \( \varphi_k \) is \( \varphi_i \varphi_j \). Next, we use the triplets to divide the factor eigenvectors into two sets that correspond to the manifolds \( M_1 \) and \( M_2 \).

3.2 Assigning the factor eigenvectors to factor manifolds

Let \( T \) be the set of triplets returned by Algorithm 1. We will use \( T \) to split the candidate factor eigenvectors into two separate subsets via an approximate MAX-CUT algorithm. To do so, we define a weighted graph \( H = (V, E, W) \) where \( V \) is the set of unique factor eigenvectors in our list of triplets produced by Algorithm 1. The undirected edge set is defined as

\[
E = \{ (i,j) \mid \exists k \text{ s.t. } (i,j,k) \in T \} \tag{20}
\]

with edge weights given by the highest similarity score,

\[
W_{ij} = \max_{(i,j,k) \in T} S(\varphi_k, \varphi_i \varphi_j). \tag{21}
\]
Sharon Zhang, Amit Moscovich, Amit Singer

Figure 2: **Rectangle with noise.** The algorithm was run on 10,000 samples with parameters $\delta = 0.5$, $\gamma = 0.85$. The eigenvectors are plotted on the ground truth $x$ and $y$ coordinate plane. **Left.** The original data, with noise in the $z$-direction indicated by color. **Middle.** First five factor eigenvectors associated with the $y$ axis as determined by the algorithm. **Right.** First five factor eigenvectors associated with the $x$ axis.

The idea here is that a high score suggests that $\varphi_i$ and $\varphi_j$ are the true factors of $\varphi_k$ and thus must belong to different factor manifolds. Using a MAX-CUT SDP solver with $H$ as the input, we sort our eigenvectors into two bins that correspond to $M_1$ and $M_2$.

### 3.3 Implementation details and runtime

The algorithm was implemented in Python, and the MAX-CUT SDP was implemented using CVXPY (Diamond and Boyd, 2016). The first part of the algorithm computes the diffusion map of the data. The eigenvectors are computed using the Scikit-learn randomized SVD implementation (Pedregosa et al., 2011; Martinsson et al., 2011). In the subsequent step, we search through $\binom{N}{3}$ triplets. Taking into account the element-wise vector multiplication at each step, this takes $O(nN^3)$ time. In practice, however, the eigenvalue criterion quickly discards of the vast majority of triplets. The last part of the algorithm iterates through the list of triplets to form an undirected weighted graph and then runs a MAX-CUT SDP solver.

### 4 Simulations

We tested our algorithm on three different datasets: noisy samples from a 2D rectangle in 3D space, a torus, and a synthetic set of cryo-electron microscopy images.

#### 4.1 2D noisy rectangle

First, we ran our algorithm on points sampled from an axis-aligned 2D rectangle $[0, \sqrt{\pi} + 1] \times [0, 1.5] \in \mathbb{R}^2$ embedded in $\mathbb{R}^3$. The data was generated by taking 10,000 random points and adding noise in the $z$-direction. We used the parameters $\gamma = 0.85$, $\delta = 0.5$. The results, shown in Figure 2, show cosines along the $x$ and $y$ directions, as expected (see Example 1).

Table 1 lists running times, averaged over 5 trials, on this dataset. A single core of a 2.9 GHz 2017 Intel Core i7 processor was used for these benchmarks.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Diffusion map</th>
<th>Alg. 1</th>
<th>Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>5.59 s</td>
<td>0.454 s</td>
<td>0.022 s</td>
</tr>
<tr>
<td>100</td>
<td>6.07 s</td>
<td>3.32 s</td>
<td>0.020 s</td>
</tr>
<tr>
<td>200</td>
<td>8.18 s</td>
<td>20.3 s</td>
<td>0.020 s</td>
</tr>
<tr>
<td>400</td>
<td>12.3 s</td>
<td>141 s</td>
<td>0.020 s</td>
</tr>
</tbody>
</table>

#### 4.2 Analysis of threshold parameters

Our algorithm has two threshold parameters: $\delta$ for the eigenvalue criterion and $\gamma$ for the eigenvector criterion. A question one might have is: for a given product eigenvector $\varphi_k$, how much of an outlier will the similarity of the true combination of factor eigenvectors $\varphi_i \varphi_j$ be compared to the similarities of all other pairs? To investigate this, for each product eigenvector $\varphi_k$ we plot the eigenvector criterion $S(\varphi_k, \varphi_i \varphi_j)$ against the eigenvalue criterion $|\lambda_i + \lambda_j - \lambda_k|$ over all $1 \leq i < j < k$, shown in Figure 4. We see that in most cases the similarity score for the chosen triplet is indeed an outlier. For product eigenvectors with lower eigenvalues, the highest similarity stands out more clearly, as this part
of the spectrum is more stable. As the eigenvalue of the product eigenvector increases, the best pair, which has the highest similarity score, is less well separated. However, this ambiguity tends to be between relatively few candidates. One can also see that the highest scoring triplet has a small eigenvalue criterion. This demonstrates that the eigenvalue criterion can speed up the triplet search without sacrificing performance.

A sensitivity analysis of both parameters is included in Figure 5. Each figure shows the manifold assignments obtained with 100 eigenvectors on the noisy rectangle dataset. The figure demonstrates the stability of the algorithm, since the assignments are consistent across a large range of parameter values.

4.3 Single-particle cryo-EM data

We present an application of manifold factorization for cryo-electron microscopy (cryo-EM). Cryo-EM is a technique used for imaging proteins and other macromolecules that has fostered major advances in structural biology (Kühlbrandt, 2014; Cressey and Callaway, 2017). Recently, it played a key role in unraveling the structural properties of the novel coronavirus SARS-CoV-2 (Zimmer, 2020). In cryo-EM, molecular samples are rapidly frozen in a thin layer of ice, thus capturing them in their native states (Dubochet et al., 1988). Once frozen, a transmission electron microscope is used to capture 2D tomographic projections of their electrostatic potential (Vulović et al., 2013). These 2D images are then used to reconstruct, through a series of computational steps, a 3D reconstruction of the molecule. (Zhou et al., 2020, Fig. 1).

The classical cryo-EM reconstruction problem is posed as follows: given \( n \) tomographic projection images \( I_1, \ldots, I_n \) of a particular molecule at random (unknown) orientations, how can we recover the three-dimensional structure of the molecule? A core obstacle to this problem is handling the low signal-to-noise ratio present in the micrographs (Singer and Sigworth, 2020; Bendory et al., 2020). This task is made even more difficult if we consider the heterogeneity problem, which has been the subject of recent literature (Frank, 2018; Nakane et al., 2018; Andén and Singer, 2018; Sorzano et al., 2019; Lederman et al., 2020; Zhong et al., 2020). Here, due to natural variations, each image \( I_i \) captures a different spatial conformation of the same molecule. Several works have applied diffusion maps for the analysis of cryo-EM datasets with continuous heterogeneity (Dashti et al., 2014; Schwaneder et al., 2014; Moscovich et al., 2020; Zelesko et al., 2020; Kileel et al., 2020; Dashti et al., 2020).

For molecules that exhibit two or more independent continuous motions, the manifold of 3D electrostatic densities is a product space. For our experiments, we use the potassium ion channel model of Moscovich et al. (2020) with two independently deformable parts: a spinning subunit with 360° of rotation, and a subunit that stretches in the \( xy \)-plane (see Figure 3). To create the images, we sampled 10,000 conformations of the molecule with uniformly drawn angles and \( x \)-axis stretches ranging from \([-20, 20]\). All the images were projected from a single view. This models the approach of Dashti et al. (2020), which computes a diffusion map for each viewing direction separately. Finally, we use PCA with four components to preprocess the raw images and transform them so that they are zero-centered with unit variance.
Figure 4: Analysis of eigenvalue and eigenvector criteria for the rectangle dataset. We picked six product eigenvectors $\phi_k$ at random. For each one, we plotted the eigenvalue criteria error ($y$ axis) and similarity scores ($x$ axis) for all triplets $(\phi_i, \phi_j, \phi_k)$, $1 \leq i < j < k$. Numbers at the top of the scatter plots are the index $k$. The point that corresponds to the eigenvector with maximum similarity is marked by a red arrow.

Figure 5: Sensitivity analysis for $\delta$ and $\gamma$. Each column corresponds to a different eigenvector and the colors denote assignment of the eigenvector to either $M_1$ (red), $M_2$ (blue) or to $M_1 \times M_2$ (white). Hence each row shows all the assignments our algorithm makes for a particular choice of the parameters. The base values of each parameter are $\gamma = 0.75$ (when varying $\delta$) and $\delta = 0.4$ (when varying $\gamma$).

We ran three different simulations: first, a dataset of molecules with rotations and stretches in the $x$ direction only and low noise; then, a dataset of molecules with rotations and stretches in the $x$-direction only and high noise; and finally, a dataset of molecules with rotations and stretches in both the $x$ and $y$ directions and low noise. The parameters $\gamma = 0.80$ and $\delta = 1.0$ were used in all simulations. The results for the second simulation are shown in Figure 3. The eigenvectors corresponding to the $x$-stretch manifold exhibit the ground truth cosine waves, and the eigenvectors corresponding to the rotation manifold (a circle) exhibit the ground truth sinusoidal waves.

Note that even though the latent space is a product space, Assumption 1 does not strictly hold due to the additive Gaussian noise. Nonetheless, our method works nicely on this dataset.

4.4 Visualizing manifold factors

We can visualize the manifold factors by plotting their diffusion map embeddings. Recall from Equation (17) that given data samples $x_1, \ldots, x_n$ and Laplacian eigenvectors $\phi_1, \phi_2, \ldots, \phi_N$, the embedding map is

$$x_i \mapsto [\phi_2(x_i), \phi_3(x_i), \ldots, \phi_\ell(x_i)].$$

For our visualizations, we use $\ell = 3$ and plot $\phi_2(x_i)$ against $\phi_3(x_i)$. In the cases where our algorithm is applied first, we can visualize the diffusion map embedding of each manifold factor. We need not perform
Figure 6: **Visualizing manifold factorizations.** We visualize three different datasets. The left two columns show the diffusion map embeddings of each manifold factor as determined by our algorithm. The middle column shows a standard diffusion map of the points from the product manifold and the rightmost column shows a three-dimensional linear ICA. The colors represent the ground truth data in a single dimension of the parameter space. **Top row.** Cryo-EM images (see Section 4.3). **Middle row.** 10,000 points uniformly drawn from a torus $S^1 \times S^1$ embedded in $\mathbb{R}^4$ where the factor circle radiuses are $2$ and $\sqrt{\pi} + 1$. **Bottom row.** 10,000 points drawn from a 2D rectangular domain with noise in the z-direction (see Section 4.1).

an additional diffusion map computation since our algorithm has already produced the eigenvectors for each manifold factor $\mathcal{M}_i$. Colors indicate the ground truth coordinate of the point $x_i$ on the respective manifold factor. The left two columns of Figure 6 show the diffusion map embeddings of manifold factors computed using our algorithm for three different experiments. In all three cases, the embeddings are in agreement with the factors of the data manifold.

While we are not aware of other dimensionality reduction methods that aim to accomplish the same goal as our method, the two most immediately comparable methods are standard diffusion maps, which our algorithm builds upon, and linear independent component analysis techniques (Linear ICA). Though these methods do not directly apply to the specific task of product manifold factorization, we can use them to visualize the entire product manifold (also shown in Figure 6).

Our algorithm can be used to accomplish several objectives. First, we can use representative factor eigenvectors from each manifold factor’s eigenspace to compress the data representation. This is a complementary approach to other methods for parsimonious representations of manifolds (Blau and Michaeli, 2017; Dsilva et al., 2018; Meila et al., 2018; Chen and Meilă, 2019). Additionally, by separating the factor eigenvectors according to their respective manifolds, we also disentangle the diffusion map coordinates into sets that correspond to independent latent factors.

5 **Hierarchical factorization**

Extending the algorithm to $m > 2$ manifolds is possible by using a MAX K-CUT SDP (Newman, 2018) in place of the MAX-CUT solver and adjusting the triplet search to an $m + 1$-tuple search. This approach requires additional evaluation and we leave it for future work. Another possibility is to run the two-factor algorithm in a hierarchical fashion, i.e. apply it again on the product eigenvectors assigned to one of the two manifolds. As an example, we ran the two-factor algorithm twice on 10,000 points drawn uniformly from the box $[0, 1.0 + \sqrt{\pi}] \times [0, 1.5] \times [0, 7.0]$. The results of the first iteration are shown in bins #1 and #2 of Figure 7. For the second iteration, we ran our algorithm on the eigenvectors of bin #2, and the resulting factor eigenvectors were sorted into bins #3 and #4. Together, the eigenvectors in bins #1, #3 and #4 form our $m = 3$ factorization.
Figure 7: **Hierarchical factorization.** Manifold factor eigenvector bins obtained by running our algorithm hierarchically on points from a 3D box (see Section 5). Bin #1 and bin #2 are obtained from the first iteration. Rerunning the factorization on the eigenvectors in bin #2 resulted in bins #3 and #4.

6 Conclusion

In this paper, we presented a spectral method for nonlinear dimensionality reduction and data representation. Our method is based on manifold factorization and applies to data sets with two or more independent degrees of freedom. We tested our algorithm on several synthetic datasets, including simulated transmission electron-microscope images of a protein with two independently moving subunits.

One direction for future work is the extension of the method to products of more than two manifolds (see Section 5). Another interesting direction is to test our approach on broader classes of spaces, other than product spaces, but for which the Laplacian operator separates nonetheless.

Reproducibility

Code for reproducing the figures in this paper can be found at: [https://github.com/sxzhang25/product-manifold-learning](https://github.com/sxzhang25/product-manifold-learning)

Acknowledgements

This work was supported in part by AFOSR Awards FA9550-17-1-0291 and FA9550-20-1-0266, ARO W911NF-17-1-0512, the Simons Foundation Math+X Investigator Award, NSF BIGDATA Award IIS1837992, NSF Award DMS-2009753, and NIH/NGMS Award R01GM136780-01. We thank Nicolas Boumal and the anonymous referees for their helpful suggestions.

References


Yochai Blau and Tomer Michaeli. Non-redundant Spectral Dimensionality Reduction. In *Joint Euro-
pean Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), 2017. doi:10.1007/978-3-319-71249-9_16.


Francesco Locatello et al. Challenging Common Assumptions in the Unsupervised Learning of Disen-


