LINEAR AND NONLINEAR DIMENSIONALITY REDUCTION

MARIA CAMERON

Contents

In this chapter, we will take a geometric view of data sets. Our goal is to discover intrinsic geometric structures in data. The key idea is that often high-dimensional data lie on a low-dimensional manifold. In this case, we would like to find the dimensionality of

this manifold. In addition, we often want to visualize the data. Then, we need to construct a map from the original space to 2D or 3D.

1. Linear dimensionality reduction

A comprehensive survey of linear dimensionality reduction methods is offered by J. Cunningham and Z. Ghahramani [1]. The abstract in this article starts with: *Linear dimensionality reduction methods are a cornerstone of analyzing high dimensional data, due to their simple geometric interpretations and typically attractive computational properties. These methods capture many data features of interest, such as covariance, dynamical structure, correlation between data sets, input-output relationships, and margin between data classes.* Linear dimensionality reduction methods are defined as:

Definition 1. *Given n D-dimensional data points combined into a matrix $X \in \mathbb{R}^{n\times D}$ and a choice of dimensionality $d < D$, optimize some objective $f_X(\cdot)$ to produce a linear transformation $P \in \mathbb{R}^{D\times d}$, and call $Y = XP \in \mathbb{R}^{n\times d}$ the low-dimensional transformed data.*

Linear dimensionality reduction methods often serve as building blocks for nonlinear ones. In the next four sections we will review four most well-known of them:

- Principal component analysis (PCA),
- Multidimensional scaling (MDA),
- Fisher's linear discriminant;
- Multiple discriminant analysis (MDA) (or linear discriminant analysis (LDA)).

2. Principal Component Analysis

Principal Component Analysis (PCA), originally formulated by Pearson (1901), is a widely used tool for data analysis in both natural and social sciences. A classical reference for the PCA is the book by I. T. Jolliffe [2] available online. Here I give just a brief overview.

Let $\eta \in \mathbb{R}^D$ be a vector random variable. Let

$$X = \begin{bmatrix} x_1^\top & \to \\ \vdots & \\ x_n^\top & \to \end{bmatrix}$$

be an $n \times D$ matrix of samples of η. The goal of the PCA is to map the samples of η from the high-dimensional space $\mathbb{R}^D$ into a low-dimensional space $\mathbb{R}^d$, while retaining as much variation present in the samples as possible.

2.1. Derivation.

We would like to replace η with a random variable $\xi \in \mathbb{R}^d$ whose components are linear combinations of the components of η,

$$\xi = (w_1^\top \eta, \dots, w_d^\top \eta)$$

where the directions w_i, $i = 1, \ldots, d$ maximize the variance $\mathsf{Var}(w_i^\top \eta)$ while satisfy the constraints

$$w_i^\top w_i = 1, \quad i = 1, \ldots, d, \quad \text{and} \quad \mathsf{Cov}(w_i\eta, w_j\eta) = 0.$$

The condition $\mathrm{Cov}(w_i\eta, w_j\eta) = 0$ means that the components of ξ should be uncorrelated.

First we find PCA 1, $\xi_1 = w_1\eta$ using Lagrange multipliers. Let

$$C := \mathsf{Cov}(\eta) = (E[(\eta_i - E[\eta_i])(\eta_j - E[\eta_j])])_{i,j=1}^{D}$$

be the covariance matrix of η. It is $D \times D$, symmetric positive definite provided that all components of η have nonzero variances. Then

$$\mathsf{Var}(\xi_1) = E[(w_1^\top \eta - w_1^\top E[\eta])^2] = w_1^\top C w_1.$$

We set up a constrained optimization problem

$$J(w_1) = \frac{1}{2} w_1^\top C w_1 \ \to \ \max, \quad \text{subject to} \quad \|w_1\|^2 = 1.$$

The Lagrangian function is given by

$$L(w_1, \lambda) = \frac{1}{2} w_1^\top C w_1 - \frac{\lambda}{2}(w_1^\top w_1 - 1),$$

where λ is the Lagrange multiplier. Differentiating L with respect to w_1 we get

$$C w_1 - \lambda w_1 = (C - I\lambda) w_1 = 0.$$

Hence w_1 must be an eigenvector of C corresponding to the eigenvalue λ. To decide which eigenvalue we pick, we recall that we are maximizing

$$w_1^\top C w_1 = \lambda w_1^\top w = \lambda.$$

Hence, we pick λ_1, the largest eigenvalue of C, and the corresponding eigenvector w_1.

Next we will look for w_2. The zero covariance condition gives:

$$\mathrm{Cov}(w_1\eta, w_2\eta) = w_1^\top C w_2 = \lambda_1 w_1^\top w_2 = 0.$$

Hence w_2 must be orthogonal to w_1. The optimization problem for w_2 is

$$J(w_2) = \frac{1}{2} w_2^\top C w_2 \ \to \ \max, \quad \text{subject to} \quad \|w_2\|^2 = 1, \quad w_1^\top w_2 = 0.$$

The Lagrangian function is

$$L(w_2, \lambda, \phi) = \frac{1}{2} w_2^\top C w_2 - \frac{\lambda}{2}(w_2^\top w_2 - 1) - \phi w_1^\top w_2.$$

Differentiating it with respect to w_2 we get:

$$C w_2 - \lambda w_2 - \phi w_1 = (C - I\lambda) w_2 - \phi w_1 = 0.$$

Taking a dot product of this equation with w_1 we get: $0 - 0 - \phi = 0$ which forces ϕ to be zero. Hence, λ and w_2 must be an eigenpair corresponding to the second largest eigenvalue of C.

Proceeding in a similar manner, we find that w_k is the eigenvector of C corresponding to its kth largest eigenvalue.

The variables $\xi_i = w_i^\top \eta$ where w_i, $i = 1, \dots, d$ are the eigenvectors of the covariance matrix C corresponding to the d largest eigenvalues, are called *the principal components*. The vector w_i, $i = 1, \dots, d$, is called *the vector of coefficients or loadings for the ith principal component*.

2.2. **Calculation in practice.** In practice, when we are dealing with data, the data points $x_i \in \mathbb{R}^D$ are interpreted as samples of a vector random variable. The covariance matrix is not known. To approximate it, we compute the $D \times D$ data covariance matrix. First we need to center the data so that column means are all zero:

$$Y := X - \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \left[\frac{1}{n} \sum_{i=1}^{n} x_{i1}, \dots, \frac{1}{n} \sum_{i=1}^{n} x_{iD} \right].$$

Then the covariance matrix is given by

$$C := \frac{1}{n} Y^\top Y.$$

Its eigendecomposition with eigenvalues ordered in the decreasing order is

$$C = W \Lambda W^\top, \quad \text{where} \quad \Lambda = \operatorname{diag}\{\lambda_1, \dots, \lambda_D\}$$

and $\lambda_1 \geq \dots \geq \lambda_D \geq 0$. The d eigenvectors corresponding to the d largest eigenvalues will be the desired loadings. The coordinates of the data points in the principal component space will be $z_i := Y w_i$, $i = 1, \dots, d$.

Now let us connect the PCA with the SVD. Let

$$Y = U \Sigma W^\top$$

be an SVD of Y. Then

$$Y^\top Y = W \Sigma^2 W^\top \equiv NC = NW \Lambda W^\top.$$

Hence the first d columns of $W = [w_1, \dots, w_D]$ are the vectors of coefficients for the first d principal components, and the principal components are $Y w_i$.

3. Multidimensional scaling

Multidimensional scaling (MDS)[3] aims at finding a low-dimensional representation of data given an $n \times n$ matrix Δ of squared distances between the data points. As we will show below, if the distances are Euclidean, MDS is equivalent to PCA. However, in many applications the distances are non-Euclidean. For example, the distances between remote cities are non-Euclidean and the cities are located on the Earth. In social sciences, the distances are often obtained from perceptual data and hence non-Euclidean (see examples in [3]). Matlab has a built-in function `cmdscale` and provides an example with a 2D embedding of some US cities from their pairwise distances.

Let us describe the classic multidimensional scaling algorithm. Input: the matrix Δ of pairwise squared distances and the desired output dimension d.

(1) Compute row means of the matrix Δ:

$$\mu_i := \frac{1}{n}\sum_{j=1}^{n}\Delta_{ij}, \quad i = 1, \dots, n.$$

(2) Set the mean of row means:

$$\mu := \frac{1}{n}\sum_{i=1}^{n}\mu_i.$$

(3) Define the $n \times n$ matrix $B = (B_{ij})$ by

$$B_{ij} := \frac{1}{2}\left(\mu_i + \mu_j - \Delta_{ij} - \mu\right).$$

(4) Compute the eigendecomposition of B, $B = V\Lambda V^\top$ where the eigenvalues λ_i are ordered in the decreasing order. If the d largest eigenvalues are positive, take the corresponding eigenpairs (v_i, λ_i), $i = 1, \dots, d$. Otherwise, take the set of m_+ eigenpairs corresponding to positive eigenvalues. Set $d_+ := \min\{m_+, d\}$ where m_+ is the number of positive eigenvalues.

(5) Define the embedding of the dataset into $\mathbb{R}^{d_+}$ by

$$Z = \begin{bmatrix} v_1, \dots v_{d_+} \end{bmatrix} \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_{d_+}} \end{bmatrix}.$$

The matrix Z is $n \times d_+$. Row i of Z corresponds to the embedding of the ith data point into $\mathbb{R}^{d_+}$.

If the distance is Euclidean, the MDS is equivalent to the PCA. Indeed, let Y be an $n \times D$ matrix of data centered so that the column means are zeros. Then

$$\Delta_{ij} = \sum_{k=1}^{D}(y_{ik} - y_{jk})^2 = \sum_{k=1}^{D} y_{ik}^2 + \sum_{k=1}^{D} y_{jk}^2 - 2\underbrace{\sum_{k=1}^{D} y_{ik}y_{jk}}_{(YY^\top)_{ij}}.$$

Hence

$$(YY^\top)_{ij} = \frac{1}{2}\left(\sum_{k=1}^{D} y_{ik}^2 + \sum_{k=1}^{D} y_{jk}^2 - (\Delta)_{ij}\right)$$

are the matrix elements of $YY^\top$. Let us show that the matrix elements of the matrix B defined in Step (3) of the MDS coincide with those of $YY^\top$. First we expand the row

means of B:

$$\begin{aligned}\mu_i &= \frac{1}{n}\sum_{j=1}^{n}\Delta_{ij} = \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{D}(y_{ik}-y_{jk})^2 \\ &= \sum_{k=1}^{D} y_{ik}^2 + \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{D} y_{jk}^2 - 2\frac{1}{n}\sum_{k=1}^{D} y_{ik}\underbrace{\sum_{j=1}^{n} y_{jk}}_{=0} \\ &= \sum_{k=1}^{D} y_{ik}^2 + \underbrace{\frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{D} y_{jk}^2}_{\|Y\|_F^2}.\end{aligned}$$

Here we have used the fact that the column means of Y are zeros. Now we expand μ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{D} y_{ik}^2 + \frac{1}{n}\|Y\|_F^2\right) = \frac{2}{n}\|Y\|_F^2.$$

Plugging in these expressions for μ_i, μ_j, and μ into the formula for B_{ij} we get:

$$\begin{aligned}B_{ij} &= \frac{1}{2}\left(\mu_i + \mu_j - \Delta_{ij} - \mu\right) \\ &= \frac{1}{2}\left(\sum_{k=1}^{D} y_{ik}^2 + \frac{1}{n}\|Y\|_F^2 + \sum_{k=1}^{D} y_{jk}^2 + \frac{1}{n}\|Y\|_F^2 - \frac{2}{n}\|Y\|_F^2 - (\Delta)_{ij}\right) \\ &= \frac{1}{2}\left(\sum_{k=1}^{D} y_{ik}^2 + \sum_{k=1}^{D} y_{jk}^2 - (\Delta)_{ij}\right) = (YY^\top)_{ij}.\end{aligned}$$

Now, if

$$Y = P\Sigma Q^\top$$

is an SVD of Y, then

$$YY^\top = P\Sigma^2 P^\top,$$

hence $\Lambda = \Sigma^2$ and $V = P$. Now,

$$Y[q_1, \dots q_d] = P\Sigma Q^\top[q_1, \dots q_d] = [p_1, \dots, p_d]\begin{bmatrix}\sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_{d_+}}\end{bmatrix},$$

which coincides with the output of the MDS.

Remark If the squared distances forming the matrix Δ are non-Euclidean, the matrix B might have some negative eigenvalues. For example in the Matlab's example mapping USA cities, some eigenvalues are negative.

Remark If the number of data points is large, the computation of the MDS becomes expensive: its cost scales as $O(n^2)$. To fight this issue the so-called landmark multidimensional scaling (LMDS) has been introduced [4] with cost $O(ln)$ where $l \ll n$ is the number of landmarks.

4. Fisher's linear discriminant

Reference: Duda, Hart, Stork "Pattern Classification" [5] (note: the GitHub version that you can google out does not contain this material).

Fisher's linear discriminant seeks a projection of a dataset consisting of two classes into a 1D space that discriminates the classes the most. Let $X \in \mathbb{R}^{n\times D}$ be a dataset consisting of two subsets, $X_1 \in \mathbb{R}^{n_1\times D}$ belonging to class 1 and $X_2 \in \mathbb{R}^{n_2\times D}$ belonging to class 2. The row indices of X belonging to class i form an index set $\mathcal{I}_i$, $i = 1, 2$. The set of projections of data points x to a 1D space is given by:

$$y_i = w^\top x_i, \quad i = 1, \dots, n.$$

Let us find $w \in \mathbb{R}^D$, $\|w\| = 1$, that discriminates the classes the most. This means that the means of the projected data should be separated as much as possible, while the projected data of each class should be clustered around the corresponding mean as much as possible. In order to formulate the corresponding optimization problem, we will find the means and the scatters for each class.

The mean of projected data from class i, $i = 1, 2$, is given by

$$\tilde{m}_i = \frac{1}{n_i}\sum_{k\in\mathcal{I}_i} y_k = \frac{1}{n_i}\sum_{k\in\mathcal{I}_i} w^\top x_k = w^\top m_i, \quad \text{where} \quad m_i = \frac{1}{n_i}\sum_{k\in\mathcal{I}_i} x_k.$$

the scatter matrix of class i is given by

$$S_i := \sum_{k\in\mathcal{I}_i}(x_k - m_i)(x_k - m_i)^\top \equiv X_{\mathcal{I}_i,:}X^\top_{\mathcal{I}_i,:}.$$

We define the *within-class* scatter matrix as the sum

$$S_w := S_1 + S_2.$$

The scatters of the projected data of each class i are:

$$\tilde{s}_i := \sum_{k\in\mathcal{I}_i}(y_k - \tilde{m}_i)^2 = w^\top \sum_{k\in\mathcal{I}_i}(x_k - m_i)(x_k - m_i)^\top w = wS_iw.$$

Therefore, the *within-class scatter* of the projected data is

$$\tilde{s}_1 + \tilde{s}_2 = w^\top S_w w.$$

The distance between the projected means is

$$|\tilde{m}_1 - \tilde{m}_2|^2 = |w^\top(m_1 - m_2)|^2 = w^\top(m_1 - m_2)(m_1 - m_2)^\top w \equiv w^\top S_b w,$$

end where S_b is called the *between-class* scatter matrix.

Since we want to maximize $w^\top S_b w$ relative to $w^\top S_w w$, we set the objective function

$$J(w) = \frac{w^\top S_b w}{w^\top S_w w}. \tag{1}$$

Note that $J(w)$ depends only on the direction of w but not on its magnitude. To maximize J, wee take its gradient:

$$\nabla J = \frac{2S_b w}{w^\top S_w w} - \frac{w^\top S_b w}{(w^\top S_w w)^2} 2S_w w = \frac{2}{w^\top S_w w}\left[S_B w - J(w) S_w w\right] = 0.$$

The last expression means that the desired direction w is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem:

$$S_b w = \lambda S_w w \quad \text{or, equivalently} \quad S_w^{-1} S_b w = \lambda w. \tag{2}$$

Furthermore, since S_b is a rank-1 matrix, as $S_b = (m_1 - m_2)(m_1 - m_2)^\top$, the vector $S_b w$ is parallel to $m_1 - m_2$. Hence, we readily set

$$w = S_w^{-1}(m_1 - m_2) \quad \text{and normalize:} \quad w \mapsto \frac{w}{\|w\|}.$$

5. Multiple discriminant analysis

Reference: [5]. Multiple discriminant analysis (MDA) is a generalization of Fisher's linear discriminant to the case of more than two classes. We assume that the dataset consists of data coming from $c > 1$ classes. Let $\mathcal{I}_i$ denote the set of indices of data from class i, $i = 1, \ldots, c$. We are seeking a projection to at most $(c-1)$-dimensional space

$$y_i = W^\top x_i,$$

such that the projected class means are separated as much as possible relative to class scatters. As before, we have:

$$\tilde{m}_i = \frac{1}{n_i} \sum_{k \in \mathcal{I}_i} y_k = W^\top m_i,$$

$$\tilde{S}_i = \sum_{k \in \mathcal{I}_i} (y_k - \tilde{m}_i)(y_k - \tilde{m}_i)^\top = W^\top S_i W.$$

The within-class scatter matrix is straightforwardly generalized:

$$S_w := \sum_{i=1}^{c} S_i.$$

The generalization of the between-class scatter matrix is less straightforward. We define S_b as the difference between the total scatter matrix $S_t := X^0 (X^0)^\top$ where X^0 denotes the centered data matrix (i.e., column sums of X^0 are zeros) and S_w. We calculate:

$$S_t = \sum_{i=1}^{n} (x_i - m)(x_i - m)^\top = \sum_{i=1}^{c} \sum_{k \in \mathcal{I}_i} (x_i - m_i + m_i - m)(x_i - m_i - m_i + m)^\top,$$

where

$$m = \frac{1}{n}\sum_{i=1}^{c} n_i m_i$$

is the overall mean. We continue:

$$S_t = \sum_{i=1}^{c}\left[\sum_{k\in\mathcal{I}_i}(x_i - m_i)(x_i - m_i)^\top + 2(m_i - m)^\top \underbrace{\sum_{k\in\mathcal{I}_i}(x_i - m_i)}_{=0} + n_i(m_i - m)(m_i - m)^\top\right]$$
$$= S_w + \sum_{i=1}^{c} n_i(m_i - m)(m_i - m)^\top.$$

Therefore, we define the between-class scatter matrix as

$$S_b = \sum_{i=1}^{c} n_i(m_i - m)(m_i - m)^\top.$$

Its rank is at most $c-1$ as it is a sum of c rank-1 matrices not all of which are independent.

Similarly to Fisher's discriminant, we set the objective function

$$J(w) = \frac{w^\top S_b w}{w^\top S_w w}.$$

There are at most $c-1$ nonzero eigenvalues. We define the projection by picking the eigenvectors corresponding to the top $d \le \mathsf{rank}(S_b)$ eigenvalues.

6. Isomap

Isomap (Tenenbaum, de Silva, and Langford, 2000) [6] is one of the first nonlinear dimensionality reduction techniques introduced. It computes a globally optimal solution for a low-dimensional embedding and is guaranteed to converge asymptotically to the true structure. The idea behind the isomap is to construct a graph by connecting near data points, compute the all-to-all distance matrix for this graph, and then use the MDS to embed it to the desired low-dimensional space. Let us workout details of the isomap algorithm and illustrate it on the Swiss Roll example.

Input. The input for the isomap is the data matrix $X \in \mathbb{R}^{n\times D}$, i.e. n data points each of which is d-dimensional, and the desired output dimension d.

For the Swiss Roll example, we generate an input by sampling n points (x_i, y_i) uniformly distributed in a rectangle, and then subjecting them to the Swiss Roll map:

$$(x, y) \mapsto (x\cos x, y, x\sin x).$$

I used the following parameters:

```
function MakeSwissRollData()
n = 1600;
data2 = rand(n,2)*[12,0;0,6] + 6;
```

```
%% Convert from 2D to 3D with the Swiss Roll mapping (x,y)->(xcosx,y,xsinx)
X = data2(:,1);
Y = data2(:,2);
data3 = [X.*cos(X) Y X.*sin(X)];
figure();
hold on;
plot3(data3(:,1),data3(:,2),data3(:,3),'.','Markersize',15,'color','b');
daspect([1,1,1])
set(gca,'Fontsize',16);
view(3);
save('SwissRollData.mat','data3');
end
```

Step 1: Construct a graph G by connecting data points x_i and x_j if

- **either** the Euclidean distance between x_i and x_j is closer than some chosen threshold ϵ (ϵ-*isomap*),
- **or** x_j is one of the k nearest neighbors of x_i for some chosen number k (k-*isomap*).

I implemented the k-isomap with $k = 10$:

```
dat = load('SwissRollData.mat');
X = dat.data3;
[n,d] = size(X);
%% compute pairwise distances
d = zeros(n);
e = ones(n,1);
for i = 1 : n
    d(i,:) = sqrt(sum((X - e*X(i,:)).^2,2));
end
%% k-isomap
% STEP 1: find k nearest neighbors and define weighted directed graph
k = 10; % the number of nearest neighbors for computing distances
% for each point, find k nearest neighbors
ineib = zeros(n,k);
dneib = zeros(n,k);
for i = 1 : n
    [dsort,isort] = sort(d(i,:),'ascend');
    dneib(i,:) = dsort(1:k);
    ineib(i,:) = isort(1:k);
end
figure();
hold on;
plot3(X(:,1),X(:,2),X(:,3),'.','Markersize',15,'color','b');
daspect([1,1,1])
for i = 1 : n
```

```
    for j = 1 : k
        edge = X([i,ineib(i,j)],:);
        plot3(edge(:,1),edge(:,2),edge(:,3),'k','Linewidth',0.25);
    end
end
set(gca,'Fontsize',fsz);
view(3);
```

The resulting graph is shown in Fig. 1(a).

Step 2: Compute shortest paths in the graph G from each vertex to all other vertices and obtain the distance matrix $\Delta = (\delta_{ij})_{i,j=1}^{n}$. Since the fact that j is among the k nearest neighbors of i does not imply that i is among the k nearest neighbors of j, we symmetrize the weight matrix of the graph. All-to-all shortest distance matrix can be computed using Dijkstra's shortest path algorithm. This algorithm is implemented in Matlab in the command `graphshortestpath`. In the code below, I make the required input for this command: sparse matrix G and the source. I also randomly select a start point `m` and an end point `mf` and display the shortest path between them. The graph is colored according to the distance from `m`.

```
% STEP 2: compute shortest paths in the graph
D = zeros(n);
ee = ones(1,k);
g = ineib';
g = g(:)';
w = dneib';
w = w(:)';
G = sparse(kron((1:n),ee),g,w);
G = G+abs(G-G');
m = randi(n);
mf = randi(n);
c = zeros(n,3);
for i = 1 : n
    [dist,path,~] = graphshortestpath(G,i);
    D(i,:) = dist;
    if i == m
        figure()
        hold on
        dmax = max(dist);
        N = 1000;
        col = parula(N);
        for ii = 1 : n
            c(ii,:) = col(getcolor(dist(ii),dmax,N),:);
            plot3(X(ii,1),X(ii,2),X(ii,3),'.','Markersize',15,'color',c(ii,:));
        end
```

```
        p = path{[mf]};
        for j = 2 : length(p)
            I = [p(j-1),p(j)];
            plot3(X(I,1),X(I,2),X(I,3),'Linewidth',2,'color','r');
        end
        view(3)
        daspect([1,1,1])
        set(gca,'Fontsize',fsz);
    end
end
```

The resulting colored graph and a path in it is displayed in Fig. 1 (b).

Step 3: use MDS to do embedding to $\mathbb{R}^d$. We use Matlab's command `mdscale` to do MDS embedding.

```
% STEP 3: do MDS
% symmetrize D
D = 0.5*(D + D');
Y = mdscale(D,2);
figure();
hold on
for ii = 1 : n
    plot(Y(ii,1),Y(ii,2),'.','Markersize',15,'color',c(ii,:));
end
% plot edges
for i = 1 : n
    for j = 1 : k
        edge = Y([i,ineib(i,j)],:);
        plot(edge(:,1),edge(:,2),'k','Linewidth',0.25);
    end
end
% plot path
for j = 2 : length(p)
    I = [p(j-1),p(j)];
    plot(Y(I,1),Y(I,2),'Linewidth',2,'color','r');
end
set(gca,'Fontsize',fsz);
daspect([1,1,1]);
```

The resulting emdedding is shown in Fig. 1 (c).

A weakness of the isomap algorithm is that the approximation of the geodesic distance is not robust to noise perturbation. I reduced the number of samples from 1600 to 800 and added Gaussian noise to X:

```
noisestd = 0.8;
X = X + noisestd*randn(size(X)); % perturb by Gaussian noise
```

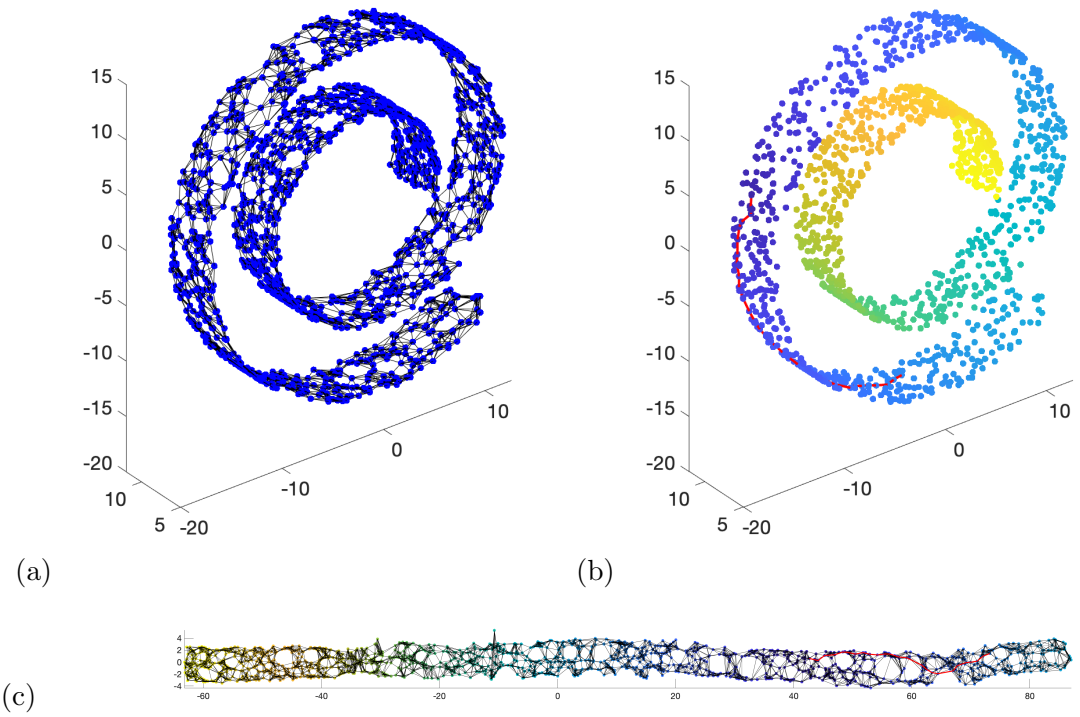

FIGURE 1. The k-isomap applied to Swiss Roll data. (a): The graph G. (b): The distances from a randomly chosen point and the shortest path between it and a randomly chosen endpoint. (c): The embedding.

The result is shown in Fig. 2. Apparently, isomap failed to recognize that the data lie on a 2D manifold just perturbed by noise.

7. Locally linear embedding (LLE)

The locally linear embedding algorithm (LLE) was proposed by S. Roweis and L. Saul (2000) [7]. Unlike the isomap requiring the calculation of all-to-all shortest paths, the LLE utilizes only local information to recover the global structure. It involves solving two linear least squares optimization problems. Sam Roweis who tragically died in January 2010 at the age of 37, left a web site on his research. In particular, there is a page devoted to the LLE with a description of the algorithm and Matlab codes. Below we elaborate the LLE algorithm with k nearest neighbors.

Input:

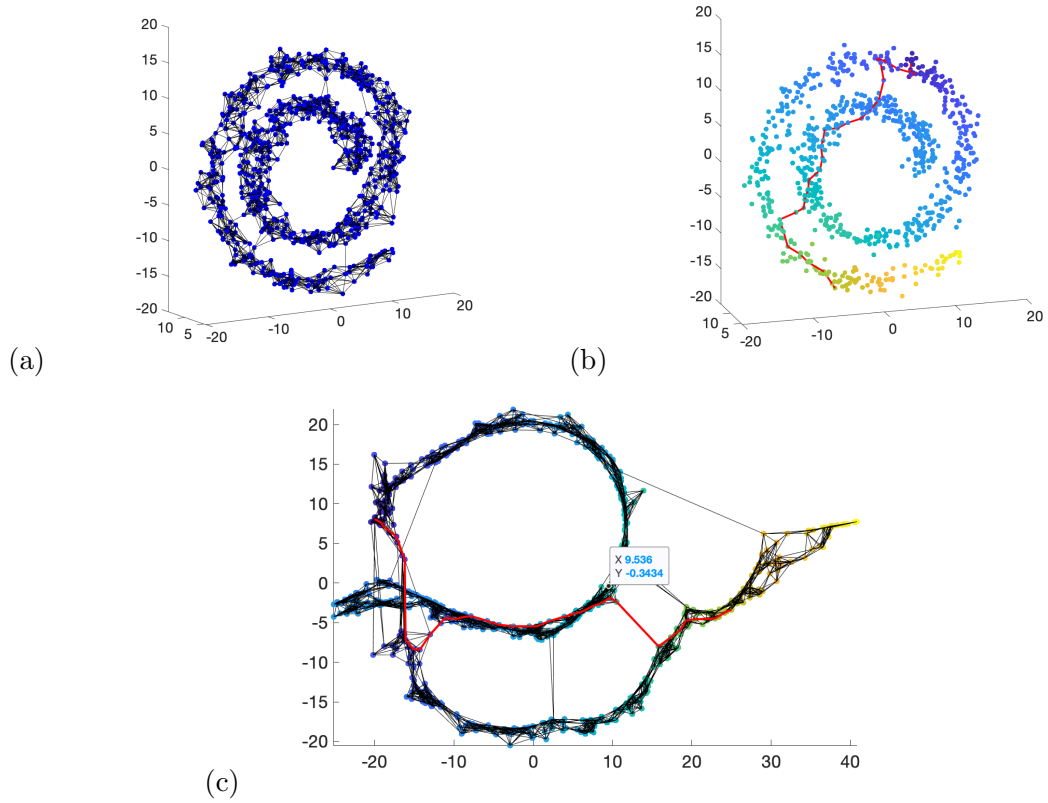

FIGURE 2. The k-isomap applied to Swiss Roll data perturbed by Gaussian noise with mean 0 and standard deviation 0.8. (a): The graph G has some undesirable interconnections. (b): The distances are messed up by these extra shortcuts. (c): The embedding is not what we would like to have.

- A $D \times n$ matrix X, whose columns represent the data points in $\mathbb{R}^D$. Note that his is the transpose of the data matrix we used before. The transposition is done for arithmetic convenience.
- The size on nearest neighborhood k. Alternatively, one can use ϵ-ball nearest neighborhood – the same two options as for the isomap.
- The desired output dimension d.

Step 1: select k nearest neighbors for each point. For each data point x_i, the $D \times k$ matrix of its nearest neighbors will be denoted by $X_{\mathcal{N}(i)}$.

Step 2: find weights for representing each data point as a linear combination of its nearest neighbors subject to constraint that their sum must be 1. Let us fix i and find the weight vector $w \in \mathbb{R}^k$ for it. This vector is the solution to the following constrained least

squares problem:

$$\min_w \frac{1}{2}\|x_i - X_{\mathcal{N}(i)}w\|^2 \quad \text{subject to} \quad 1_{1\times k}w = 1. \tag{3}$$

Since the vector w must sum up to 1, the objective function can be rewritten as

$$\frac{1}{2}\|x_i - X_{\mathcal{N}(i)}w\|^2 = \frac{1}{2}w^\top \left(x_i 1_{1\times k} - X_{\mathcal{N}(i)}\right)^\top \left(x_i 1_{1\times k} - X_{\mathcal{N}(i)}\right) w.$$

Introducing $Z := \left(x_i 1_{1\times k} - X_{\mathcal{N}(i)}\right)$, we write the Lagrangian function

$$L(w,\lambda) = \frac{1}{2}w^\top Z^\top Z w - \lambda(1_{1\times k}w - 1).$$

Its gradient with respect to w is given by

$$\nabla_w L = Z^\top Z w - \lambda 1_{k\times 1} = 0.$$

Hence w can be found by setting

$$\tilde{w} = \left(Z^\top Z\right)^{-1} 1_{k\times 1} \quad \text{and enforcing the constraint} \quad w = \frac{\tilde{w}}{1_{1\times k}\tilde{w}}.$$

Note that if $k > D$, the $k \times k$ matrix $Z^\top Z$ is singular. In this case, it is regularized by adding a small multiple of the identity matrix:

```
tol=1e-3;      % if k>D
C = z'*z;       % local covariance
C = C + eye(k,k)*tol*trace(C);    % regularlization (k>D)
```

We do it for all $i = 1 \dots n$ and obtain the weight matrix W by defining a zero $n \times n$ matrix and replacing its entries in row i corresponding to the nearest neighbors of i with the found weights for that i.

Step 3: map to embedded coordinates. The idea behind step 3 is that if the data truly lie on a low-dimensional manifold, this manifold can be approximated with a collection of local charts (neighborhoods) which are patched together to form the global manifold by rotation, translation, and rescaling. The collection of weights found in the previous step is invariant with respect to these transformations. Therefore, we expect that these weights are equally valid to represent the original manifold in $\mathbb{R}^D$ and the "unfolded" manifold in $\mathbb{R}^d$. Hence, to find the desired manifold, we set up and solve the following least squares problem

$$\min_Y \frac{1}{2}\|Y - WY\|_F^2 \quad \text{subject to} \quad 1_{1\times n}Y = 0_{1\times d}, \quad \frac{1}{n}Y^\top Y = I_{d\times d}. \tag{4}$$

Here Y is the $n \times d$ matrix of data mapped to $\mathbb{R}^d$, W is the $n \times n$ weight matrix obtained in the previous step. The constraint $1_{1\times n}Y = 0_{1\times d}$ forces the embedding to be centered at the origin. The constraint $\frac{1}{n}Y^\top Y = I_{d\times d}$ is imposed to avoid degenerate solutions. The solution to (4) is given by the bottom $d+1$ eigenvectors of

$$M := (I - W)^\top (I - W).$$

We discard the unit eigenvector corresponding to the eigenvalue 0. Indeed, the matrix W has row sums 1, hence $(I-W)1_{n\times 1}=0$. The bottom eigenvectors from 2 to $d+1$ give the desired embedding.

Let us clarify this. We will compute columns of $Y=[y_1,y_2,\dots,y_d]$ one-by-one to solve the constrained minimization problem (4):

$$\begin{aligned}
&\|(I-W)y_0\|_2^2\to\min, \quad \|y_0\|_2^2=n,\\
&\|(I-W)y_1\|_2^2\to\min, \quad \|y_1\|_2^2=n, \quad y_0^\top y_1=0,\\
&\|(I-W)y_2\|_2^2\to\min, \quad \|y_2\|_2^2=n, \quad y_j^\top y_2=0,\ j=0,1\\
&\qquad\qquad\vdots\\
&\|(I-W)y_d\|_2^2\to\min, \quad \|y_d\|_2^2=n, \quad y_j^\top y_d=0,\ j=0,1,\dots,d-1.
\end{aligned}$$

Solving this system we find that $y_0=1_{n\times 1}$ corresponding to eigenvalue 0 of M. Since M is symmetric, its eigenvectors are orthogonal. Therefore, all eigenvectors of M starting for y_1 are orthogonal to y_0. Hence, the condition $1_{1\times n}Y=0_{1\times d}$ holds. Then, the solution to

$$\|(I-W)y_1\|_2^2\to\min, \quad \|y_1\|_2^2=n, \quad y_0^\top y_2=0$$

is the eigenvector y_1 with appropriate normalization corresponding to the smallest positive eigenvalue of M. the solution to

$$\|(I-W)y_2\|_2^2\to\min, \quad \|y_2\|_2^2=n, \quad y_j^\top y_2=0,\ j=0,1$$

is the eigenvector y_2 with appropriate normalization corresponding to the second smallest positive eigenvalue of M. And so on. The first d rows of Y give the desired embedding.

An application of this algorithm to a Swiss Roll data is shown in Fig. 3.

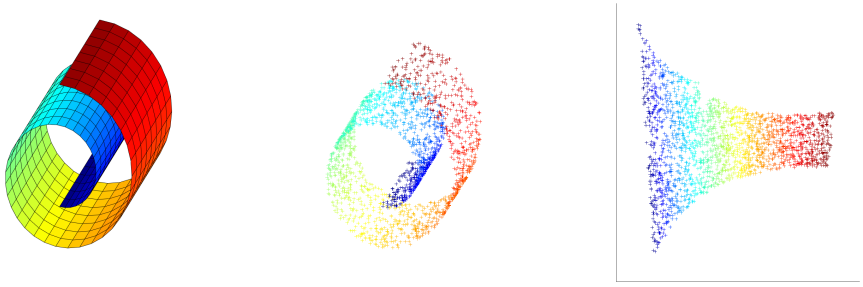

FIGURE 3. An application of the LLE algorithm to a Swiss Roll data. Left: the Swiss Roll. Middle: the input data. Right: the embedding to 2D obtained by the LLE algorithm. This figure is made using S. Roweis's code.

8. t-distributed stochastic neighbor embedding (t-SNE)

The technique called t-SNE (L. van der Maarten and G. Hinton, 2008 [8]) is a powerful nonlinear tool for data visualization, organization, and clustering. It was shown in [8] that it is superior to the isomap, LLE, its predecessor SNE (stochastic neighbor embedding), and some other nonlinear tools. In particular, it almost perfectly clusters the MNIST dataset of handwritten digits (Fig. 2(a) in [8]). Here is a web page for t-SNE maintained by L. van der Maarten. t-SNE is implemented in the Matlab command `tsne`.

8.1. **The predecessor: SNE.** t-SNE had a predecessor technique called stochastic neighbor embedding (SNE) by G. Hinton and S. Roweis (2002) [9]. SNE was motivated by the need for placing ambiguous data points (such as the word bank that can be related to finance or river) without forcing its multiple home clusters to be imaged close to each other. This technique featured the use of Gaussian kernels to design probability distributions $P = (p_{ij})$ in the original space and $Q = (q_{ij})$ in the embedding space, and the Kullback-Leibler divergence (a non-symmetric measure of the difference between two probability distributions) as the cost function function to optimize the placement of the images in the embedding space. While SNE had involved beautiful and promising ideas, it suffered from two problems. First, the cost function in it was hard-to-optimize. Optimization was taking hours and required lots of adjustments of parameters. Second, it tended to create crowding in the middle of the embedding space as the repulsive forces in its cost function were very weak in the case of placing originally far objects close in the embedding space.

8.2. **Background: entropy, information, Kullback-Leibler divergence.** The construction of t-SNE borrows some ideas from the information theory. Below we will go over some concepts useful for understanding the construction of the SNE and the t-SNE. In addition, here is a very elementary article by V. Kulkarni on entropy and cross-entropy oriented toward data scientists that might be helpful.

8.2.1. *Entropy of an information source.* Let X be a discrete random variable, and p be its *probability mass function (pmf)*. Imagine that X is a message, then $p(x)$ is the probability that $X = x$. We want to evaluate the expected gain of information upon receiving the message. If the message is known in advance, i.e., the probability space consists of a single point assumed with probability 1, no information is gained upon receiving the message. On the other hand, if X takes one of n possible values with equal probability, then the expected information gain upon receiving the message is maximal. The function

$$H(p) = -\sum_i p_i \log p_i \tag{5}$$

possesses exactly this property. $H = 0$ if the probability mass is concentrated at a single outcome. On the other hand, the uniform distribution maximizes H. Indeed, the Lagrangian function is given by:

$$L(p, \lambda) = -\sum_i p_i \log p_i - \lambda \left(\sum_i p_i - 1 \right). \tag{6}$$

Taking its gradient with respect to p and setting it to zero we get:

$$\frac{\partial}{\partial p_i} L(p, \lambda) = 1 - \log p_i - \lambda = 0, \quad i = 1, \dots n.$$

Hence, for all i, $p_i = \exp(1-\lambda)$. Since the probability mass function sums up to 1, $p_i = 1/n$, and $\lambda = 1 - \log n$.

The Shannon entropy of a distribution p measured in bits or nats is defined, respectively, by

$$H(p) = -\sum_i p_i \log_2 p_i \quad \text{or} \quad H(p) = -\sum_i p_i \log p_i. \tag{7}$$

Note that if X is the set of all possible binary sequences of length n, each of which has equal probability 2^{-n}, then

$$H = -\sum_{i=1}^{2^n} 2^{-n} \log_2(2^{-n}) = n,$$

the length of the sequences, i.e., the number of information bits.

The Shannon entropy generalized for continuous random variables is called the differential entropy:

$$h(f) = -\int f(x) \log f(x) dx, \quad \text{where} \quad f(x) \text{ is the pdf of } X. \tag{8}$$

One can prove a theorem that for a given variance σ^2, the Gaussian random variable $X \sim \mathcal{N}(m, \sigma^2)$ has the maximal entropy. A proof can be conducted e.g. using calculus of variation – see wiki.

Thus, the Shannon entropy of a random variable X is a measure of uncertainty associated with X if only its distribution is known. For a random variable, discrete with pmf p or continuous with pdf p, the Shannon entropy can be expressed as

$$H(p) = -\mathbb{E}_p[\log p]. \tag{9}$$

8.3. **Cross-entropy.** Let X and Y be two random variables defined over the same set of outcomes, and p and q be their pmfs (or pdfs), respectively. The cross-entropy of the pmf (or pdf) q relative to the pmf (or pdf) p is

$$H(p, q) = -\mathbb{E}_p[\log q]. \tag{10}$$

In words, $H(p, q)$ is a measure of uncertainty in the predicted distribution q while the true distribution is p. Let us show that for a fixed p, the cross-entropy is minimized if $q = p$. We will conduct the proof for discrete distributions. For continuous ones, the proof is similar, but the gradients are replaced with variations. We set the Lagrangian function

$$L(q, \lambda) = -\sum_i p_i \log q_i - \lambda \left(\sum_i q_i - 1 \right). \tag{11}$$

Taking its gradient with respect to q and setting it to zero we get:

$$\frac{\partial L}{\partial q_i} = -\frac{p_i}{q_i} - \lambda = 0, \quad \text{i.e.} \quad p_i = -\lambda q_i \ \forall i.$$

Since the distribution q must sum up to 1, we find that $q_i = p_i$ for all i. To check that this is a minimum, we calculate

$$\frac{\partial^2 L}{\partial q_i^2} = \frac{p_i}{q_i^2} > 0 \ \ \forall i.$$

8.4. **The Kullback-Leibler divergence.** The Kullback-Leibler (KL) divergence $D_{KL}(p||q)$ is a measure of how the probability distribution q is different from the probability distribution p. It is defined by

(12)
$$D_{KL}(p||q) = H(p,q) - H(p) = -\sum_x p(x)\log q(x) + \sum_x p(x)\log p(x) = \sum_x p(x)\log\frac{p(x)}{q(x)}.$$

The definition for continuous distributions is similar. Note that $D_{KL}(p||q) \geq 0$ and the equality takes place if and only if $p = q$ a.e..

8.5. **The construction of the t-SNE.**

8.5.1. *Probability distribution in the original space.* Let $\{x_i\}_{i\in I}$ be the original dataset with all $x_i \in \mathbb{R}^D$. For each $i \in I$, we define the conditional probability $p_{j|i}$ that x_j will be picked as a neighbor of x_i:

(13)
$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)},$$

where σ_i^2 is the variance of the Gaussian centered at x_i. The probabilities $p_{i|i}$ are set to zero for all $i \in I$. Since the data points are not distributed uniformly, it is not reasonable to use the same variance for all i. In denser regions, σ_i's should be smaller than in rarefied ones. Both SNE and t-SNE look for σ_i for each i so that the perplexity of x_i which is a smooth measure of the effective number of neighbors of x_i is equal to a user-picked number (usually between 5 and 50). The perplexity is defined by

(14)
$$\mathsf{Perp}(P_i) = 2^{H(P_i)} \quad \text{where} \quad P_i = \{p_{j|i} \mid j \in I\}, \quad H(P_i) = -\sum_{j\in I} p_{j|i}\log_2 p_{j|i},$$

i.e., P_i is the probability distribution for j being picked as a neighbor of i, and $H(P_i)$ is the Shannon entropy of this distribution measured in bits. Then σ_i is found by binary search, i.e., a bisection search in a sorted array.

Then the probability distribution is symmetrized by

(15)
$$p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n}.$$

Defined this way, p_{ij} is the probability that data points i and j are selected as neighbors. We check that

$$\sum_i \sum_j p_{ij} = \frac{1}{2n}\left(\sum_i \sum_j p_{j|i} + \sum_j \sum_i p_{i|j}\right) = \frac{n+n}{2n} = 1.$$

8.5.2. *Probability distribution in the embedding space.* The probability distribution in the embedded space is constructed using Student's t-distribution with one degree of freedom whose pdf is

$$f(t) = \frac{1}{\pi(1+t^2)}. \tag{16}$$

This distribution is picked thanks to its heavy-tails that help to fight the crowding problem in the low-dimensional embedding space. The symmetrized distribution q is defined by

$$q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k\neq l}\left(1+\|y_l - y_k\|^2\right)^{-1}}, \quad q_{ii} = 0. \tag{17}$$

Its sum over all i and j is 1. It is suggested in [8] to sample the initial guess for $\{y_i\}_{i\in I}$ from the Gaussian distribution with mean 0 and variance 10^{-4}, i.e., all y's are initially near the origin.

8.6. **Objective function: the KL divergence between q and p.** The KL divergence between q and p is picked as the objective function. The closer these distributions are, the smaller is the value of this function. Therefore, this is a quite natural choice:

$$C(Y) = D_{KL}(p||q(Y)) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}(Y)}, \quad \text{where} \quad Y = \{y_i\}_{i\in I}. \tag{18}$$

The computation of the gradient of C with respect to y is tedious but the result is simple:

$$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})\frac{y_i - y_j}{1+\|y_i - y_j\|^2}. \tag{19}$$

8.6.1. *Minimizing the objective function.* The optimization algorithm proposed in [9, 8] for optimizing $C(y)$ is the gradient descent with momentum of the form:

$$Y^{(t)} = Y^{(t-1)} + \eta\nabla C + \alpha(t)\left(Y^{(t-1)} - Y^{(t-2)}\right), \tag{20}$$

where $Y^{(t)}$ is the set of y's at iteration t, η is referred to as the *learning rate*, and $\alpha(t)$ represents the momentum at iteration t. The number of iterations T for the examples in [8] was set to 1000, and $\alpha(t) = 0.5$ for $t < 250$ and $\alpha(t) = 0.8$ for $t \geq 250$. The learning rate η was originally set to 100 and then adapted using Jacob's (1988) scheme. In addition, two more tricks were used:

- *Early compression* i.e., adding Tikhonov regularization. This forces y's to stay close together at the start of optimization. When the distances between map points are small, it is easy for clusters to move through one another so it is much easier to explore the space of possible global organizations of the data.
- *Early exaggeration* i.e., multiplying all p_{ij}'s by 4. This forces large p_{ij}'s to be modeled by large q_{ij}'s, i.e., natural clusters form tight clusters in the embedded space. This creates, in turn, a lot of empty space in the y-space and facilitates easy motion of tight clusters relative to each other. This was done for the first 50 iterations in [8].

8.7. **Example: Swiss Roll with noise.** To get sense how t-SNE works and compare it with other techniques, we apply it to the Swiss Roll example with noise. The data X used are the same as in Section Isomap, and the noise intensity is the same as in Fig. 2 where the embedding by isomap was corrupted by extra connections due to the noise:

```
dat = load('SwissRollData.mat');
X = dat.data3;
noisestd = 0.8;
X = X + noisestd*randn(size(X)); % perturb by Gaussian noise
```

The t-SNE embedding was done by the built-in Matlab function.

```
[Y,loss] = tsne(X,'Algorithm','exact','Perplexity',30,'Exaggeration',4);
fprintf('KL divergence = %d\n',loss);
```

The option `'Algorithm', 'exact'` optimizes the Kullback-Leibler divergence of distributions between the original space and the embedded space. The default 'Algorithm' is 'barneshut'. It performs an approximate optimization that is faster and uses less memory when the number of data rows is large. The option 'Exaggeration' is set to its default value. The default value of 'Perplexity' is 30. I also tried to set it to 12 which is the number of neighbors used in the LLE algorithm. The results are shown in Fig. 4. As you see, t-SNE tries to split the dataset to clusters. Decreasing perplexity leads to more clusters.

8.8. **Example: MNIST digits.** tSNE is a tool for unsupervised learning. The code below shows how one can apply it to cluster MNIST digits. The result is shown in Fig. 5. We see that the t-SNE algorithm successfully clustered the set of 10000 mnist digits.

```
close all
% load data
data = load('mnist.mat');
imgs_test = data.imgs_test;
labels_test = data.labels_test;
%%
[d1,d2,n] = size(imgs_test);
X = zeros(n,d1*d2);
for j = 1:n
    aux = squeeze(imgs_test(:,:,j));
    X(j,:) = aux(:)';
```

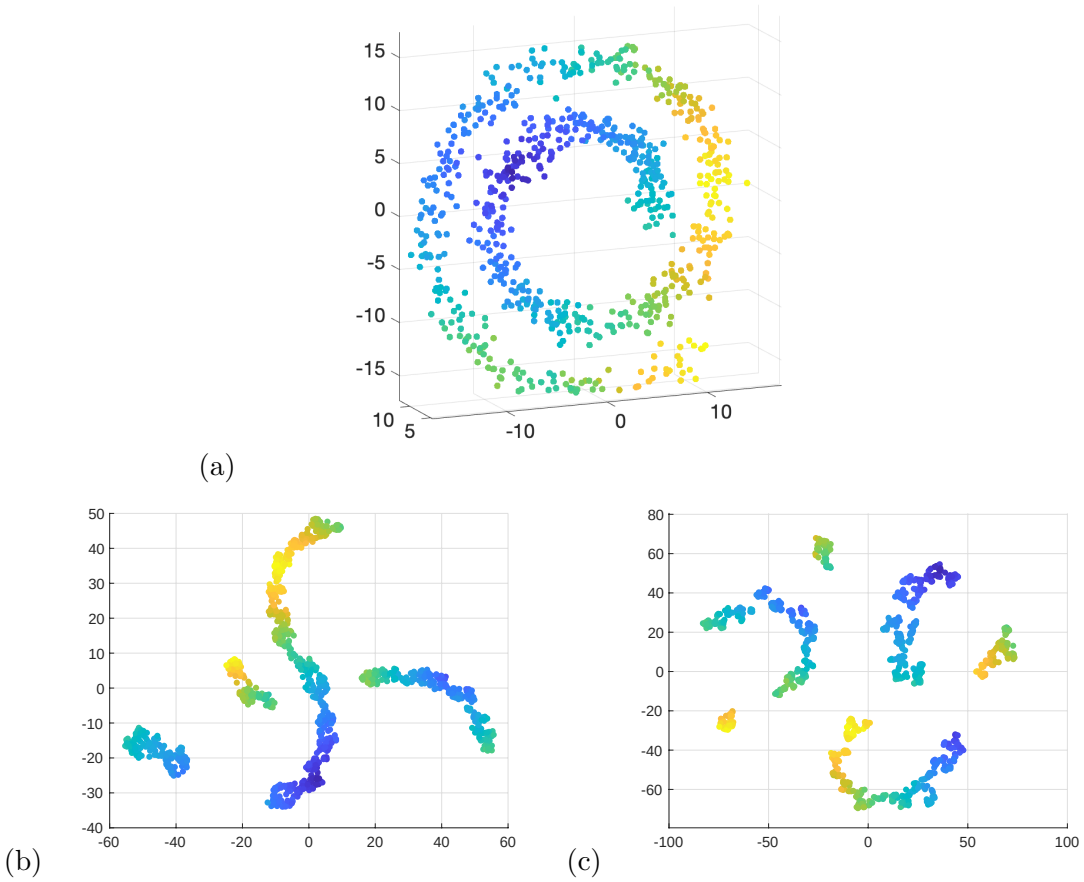

FIGURE 4. t-SNE applied to a Swiss roll data. (a): Data perturbed by Gaussian noise with standard deviation 0.8. (b): `[Y,loss] = tsne(X,'Algorithm','exact','Perplexity',30);` (c): `[Y,loss] = tsne(X,'Algorithm','exact','Perplexity',12);`

```
end
%%
[Y,loss] = tsne(X,'Algorithm','exact','Perplexity',30,'Exaggeration',4);
fprintf('KL divergence = %d\n',loss);
%%
figure;
hold on
grid
col = jet(10);
for k = 1 : 10
```

```
    ind = find(double(labels_test) == k);
    plot(Y(ind,1),Y(ind,2),'.','Markersize',20,...
     'color',col(k,:),'Displayname',sprintf('%d',k-1));
end
legend;
set(gca,'Fontsize',20)
```

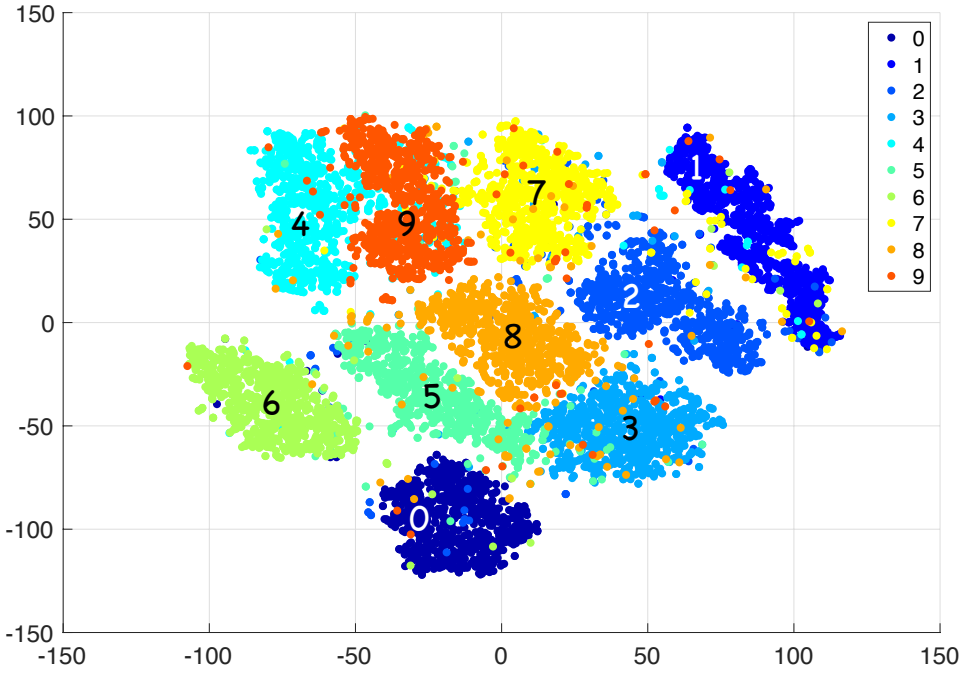

FIGURE 5. t-SNE applied to 10000 MNIST digits (the test set).

9. DIFFUSION MAPS

While the PCA is a power tool when the data points lie near a d-dimensional hyperplane in $\mathbb{R}^D$, it might fail to give a nice embedding if the data are instead located near some d-dimensional curved manifold. To handle this case, Coifman and Lafon (Yale University, 2006) introduced the so-called *diffusion maps* [10]. The key idea of this approach is to devise a discrete-time Markov chain on the data points and define the distances between remote points using the stochastic matrix of this Markov chain. This approach is robust to noisy data and is capable of adequately representing complex geometries of data structures.

This dimensional reduction technique has been successfully applied to problems arising in protein dynamics (e.g. [11, 12]). The diffusion map algorithm requires providing a bandwidth parameter ϵ whose choice is nontrivial and has been a subject of active research. One of the first approaches to tackle the problem of choosing ϵ was proposed by A. Little, M. Maggioni, and L. Rosasco. Later, simpler and more robust approaches were proposed by Lindenbaum et al. [13] and T. Berry, J. Harlim, and D. Giannakis [14, 15, 16].

9.1. **Background: properties of stochastic matrices.** An $n \times n$ matrix P is called *stochastic* if its entries are nonnegative and its row sums are equal to 1. The entries $P_{i,:}$ can be interpreted as the transition probabilities from state i: p_{ij} is the probability that the system currently at state i will go next to state j.

Definition 2. *We say that a sequence of random variables $(X_k)_{k\geq 0}$, $X_k : \Omega \to S \subset \mathbb{Z}$, is a Markov chain with initial distribution λ and stochastic matrix $P = (p_{ij})_{i,j\in S}$ if*

(1) *X_0 has distribution $\lambda = \{\lambda_i \mid i \in S\}$ and*
(2) *the Markov property holds:*

$$\mathbb{P}(X_{k+1} = i_{k+1} \mid X_k = i_k, \dots, X_0 = i_0) = \mathbb{P}(X_{k+1} = i_{k+1} \mid X_k = i_n) = p_{i_k i_{k+1}}.$$

We will write a probability distribution as a row vector. One can show that if λ is the initial probability distribution, then the probabiliby distribution after one step becomes λP, in two steps λP^2, in k steps λP^k, and so on. A probability distribution π is *invariant* if

$$\pi P = \pi \quad \text{and} \quad \sum_{i=1}^{n} \pi_i = 1.$$

If μ is a row vector with n entries such that $\mu P = \mu$, we say that μ is an *invariant probability measure*. Note that μ does not need to sum up to 1[1].

We will limit ourselves to a special kind of Markov chains arising in diffusion maps:

- The number of states is finite: $|S| = n$.
- The stochastic matrix P is irreducible and aperiodic. *Irreducibility* means that any state can be reached from any state in a finite number of jumps, i.e, for any pair i, j, $(P^t)_{ij} > 0$ for some $t \in \mathbb{N}$. *Aperiodicity* means that for any state i and for all sufficiently large t, there is a nonzero probability of returning to i in t steps, i.e, $(P^t)_{ii} > 0$ for all large enogh t and for all i. In this case, one can prove that there exists a *unique invariant probability distribution* π, and for any initial probability distribution λ we have
$$\lim_{k\to\infty} \lambda P^k = \pi.$$
- The Markov chain is time-reversible, or, equivalently, possesses the property of detailed balance: if π is the invariant probability distribution then
$$\pi_i P_{ij} = \pi_j P_{ji},$$

[1]Note that is the set of states is infinite, invariant measure may exist while invariant distribution does not exist. For example, consider a symmetric random walk on $\mathbb{Z}$.

The detailed balance means that, on average, the number of transitions from i to j per time unit is the same as that from j to i.

9.1.1. *Spectral decomposition.* The detailed balance property can be written in the matrix form:

$$\Pi P = P^\top \Pi, \quad \text{where} \quad \Pi := \mathsf{diag}\{\pi_1, \dots, \pi_n\}.$$

Note that the detailed balance condition $\Pi P = P^\top \Pi$ implies that ΠP is symmetric. Indeed, its transpose is $P^\top \Pi$. Hence, the stochastic matrix P is decomposable as

$$P = \Pi^{-1} \tilde{K}, \quad \text{where } \tilde{K} \text{ is symmetric.}$$

Furthermore, P has one eigenvalue equal to 1. The corresponding right eigenvector is $r_0 = [1, \dots, 1]^\top$ (as row sums are all 1), while the corresponding left eigenvector is π, the invariant distribution (as $\pi P = \pi$). All other eigenvalues of P are less than 1 in absolute value. The fact that they do not exceed 1 in absolute value readily follows for Gershgorin circle theorem saying that the eigenvalues of a matrix A are located within the union of Gershgorin discs $D(a_{ii}, R_i) \subset \mathbb{C}$, where $R_i = \sum_{j \neq i} |a_{ij}|$. Each such disc is centered on the real axis in the interval $[0, 1]$ and has a radius at most 1. The fact that all other eigenvalues are less than 1 in absolute value follows from aperiodicity and irreducibility.

Exercise Prove this.

The detailed balance condition $\Pi P = P^\top \Pi$ implies that P is similar to a symmetric matrix

$$\Pi^{1/2} P \Pi^{-1/2} = \Pi^{-1/2} (\Pi P) \Pi^{-1/2}.$$

Hence all eigenvalues of P are real. Furthermore, let

$$V \Lambda V^\top$$

be the spectral decomposition of $\Pi^{1/2} P \Pi^{-1/2}$. Then

$$V^\top \Pi^{1/2} P \Pi^{-1/2} V = (\Pi^{-1/2} V)^{-1/2} P (\Pi^{-1/2} V) = \Lambda.$$

Hence

$$P = (\Pi^{-1/2} V) \Lambda (\Pi^{-1/2} V)^{-1}$$

is the eigendecomposition of P. Denoting the matrix $\Pi^{-1/2} V$ of right eigenvectors of P by R, we express $V = \Pi^{1/2} R$. Hence, the matrix $L = (\Pi^{-1/2} V)^{-1} = V^\top \Pi^{1/2}$ of left eigenvectors of P is expressed via R and Π as:

$$L = V^\top \Pi^{1/2} = R^\top \Pi.$$

Hence, the eigendecomposition of P is

$$P = R \Lambda R^\top \Pi. \tag{21}$$

Since $RL = LR = I$, we have

$$R^\top \Pi R = I. \tag{22}$$

9.2. **A basic construction of a diffusion map.** First, we present the most basic diffusion map algorithm corresponding to $\alpha = 0$ in [10]. This construction is very similar to the construction of Laplacian eigenmap by Belkin and Niyogi (2003) [17].

Let $X = (x_{ik})$ be an $n \times D$ matrix of data. The rows x_i, $i = 1, \dots, n$, of X represent data points lying in $\mathbb{R}^D$.

- First, we compute the squared-distance matrix between the data points:
$$\Delta(i,j) = \sum_{k=1}^{D} (x_{ik} - x_{jk})^2.$$
- Next, we pick a scaling parameter ϵ and define the diffusion kernel, an $n \times n$ matrix $K = (k_{ij})$ where
$$k_{ij} = \exp\left(-\frac{\Delta(i,j)}{\epsilon}\right).$$
A good choice of ϵ is very important. ϵ should be comparable to squared distances from the data points to their neighbors. In practice, pick a reasonable initial guess for ϵ and then tune it experimentally. One way to pick an initial ϵ is the following. We find row minima among off-diagonal entries for the matrix Δ . Then we find the mean of these minima and set ϵ to be double this mean:
```
for i = 1 : N
    drowmin(i) = min(d(i,setdiff(1:N,i)));
end
epsilon = 2*mean(drowmin);
```
Then, if the result is not satisfactory, keep increasing the factor by which the mean of row minima is multiplied in the last line until the embedding starts making sense. This recipe is good for now. A detailed discussion on choosing ϵ is found in Section 9.7 below.
- Convert the diffusion kernel K into a stochastic matrix $P = (p_{ij})$ by dividing each row of K by the corresponding row sum:
$$P = Q^{-1}K \quad \text{where} \quad Q := \mathsf{diag}\left\{\sum_{j=1}^{n} k_{1j}, \dots, \sum_{j=1}^{n} k_{nj}\right\} := \mathrm{diag}\{q_1, \dots, q_n\}.$$
Indeed, all entries of the resulting matrix P are nonnegative, and its row sums are one.

  Note that the diagonal entries of Q constitute an invariant probability measure. Indeed:
$$[q_1, \dots, q_n]Q^{-1}K = [1, \dots, 1]K = [q_1, \dots, q_n]$$
as $K = K^\top$ and both ith row and ith column sum of K is q_i. To obtain the invariant probability distribution, we normalize $[q_1, \dots, q_n]$ so that it sums up to one:
$$\pi = \frac{q}{\sum_{i=1}^{n} q_i} \quad \text{where} \quad q := [q_1, \dots, q_n].$$

- Let us take tth power of the matrix P and denote its entries by $p_{ij}^t \equiv (P^t)_{ij}$. The entry p_{ij}^t is the probability to transition from i to j in t steps, $t \in \mathbb{N}$. A family of *diffusion distances* indexed by $t \in \mathbb{N}$ is defined by

$$\boxed{D_t(x_i, x_j)^2 := \sum_{m=1}^{n} \frac{1}{\pi_m} \left| p_{im}^t - p_{jm}^t \right|^2 .} \tag{23}$$

  Hence, the diffusion distance is a weighted l_2 distance between rows i and j of the matrix P^t. Note that $D_t(x_i, x_j)$ will be small if there is a large number of short paths connecting x_i and x_j, which makes the transition for either of them to any state x_m approximately equally likely. The power t plays the role of a scale parameter. Let us list interesting features of the diffusion distance:
  - Since it reflects the connectivity of the data at a given scale, points are closer if they are highly connected in the graph. Therefore, this distance emphasizes the notion of a cluster.
  - The quantity $D_t(x_i, x_j)$ involves summing over all paths of length t connecting x_i and x_j. This number is very robust to noise perturbation, unlike the geodesic distance.
- The family of *diffusion maps* $\Psi_t : \mathbb{R}^D \to \mathbb{R}^{n-1}$ indexed by $t \in \mathbb{N}$ from the data space $\mathbb{R}^D$ to the diffusion space $\mathbb{R}^{n-1}$ is defined so that the Euclidean distances $\|\Psi_t(x_i) - \Psi_t(x_j)\|$ in the diffusion space are equal to the diffusion distances $D_t(x_i, x_j)$.

  Let

$$P = R\Lambda L \equiv R\Lambda R^\top \Pi$$

  be the spectral decomposition of P with ordered eigenvalues:

$$1 = \lambda_0 > |\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_{n-1}|.$$

  The diffusion map Ψ_t is defined by:

$$\Psi_t(x_i) := \begin{bmatrix} \lambda_1^t r_1(i) \\ \vdots \\ \lambda_{n-1}^t r_{n-1}(i) \end{bmatrix}, \tag{24}$$

  where $R := [r_0, r_1, \ldots, r_{N-1}]$ is the matrix of right eigenvectors of P normalized so that $R^\top \Pi R = I$. Respectively, λ_m^t, is the tth power of λ_m, $m = 1, \ldots, n-1$. Note that since P is irreducible and aperiodic (as $P_{ii} > 0$, $i = 1, \ldots, n$) by construction, $\lambda_0 = 1$ and $r_0 = [1, \ldots, 1]^\top$. In Eq. (24), $r_m(i)$ denotes the ith entry of the vector r_k. In other words, $\Psi_t(x_i)$ is the transposed ith row of the matrix

$$(R\Lambda^t)^\top \equiv [\lambda_1^t r_1, \lambda_2^t r_2, \ldots, \lambda_{n-1}^t r_{n-1}]^\top .$$

  Note that we remove the first column of R because it consists of all ones and hence is not informative.

Proposition 1.

$$D_t(x_i, x_j)^2 = \sum_{m=1}^{n-1} \lambda_m^{2t} |r_m(i) - r_m(j)|^2, \tag{25}$$

i.e., the diffusion distance in the data space equals the Euclidean distance in the diffusion space.

We will prove this proposition after we finish the description of the construction.

- The diffusion maps allow us to do dimensional reduction by keeping only the first few components of $\Psi_t(\cdot)$. Often it is desirable to keep only the first two or three entries of $\Psi_t(\cdot)$ as then the diffusion map is readily visualizable. To make the dimension of the embedding space justified, we introduce an accuracy parameter $\delta \in (0, 1)$ and define the number of terms to keep:

$$s(\delta, t) = \max\{m \in \mathbb{N} \text{ such that } |\lambda_m|^t > \delta |\lambda_1|^t\}. \tag{26}$$

Then, up to relative precision δ, we have:

$$D_t(x_i, x_j)^2 = \sum_{m=1}^{s(\delta,t)} \lambda_m^{2t} |r_m(i) - r_m(j)|^2, \tag{27}$$

and

$$\Psi_t(x_i) = \begin{bmatrix} \lambda_1^t r_1(i) \\ \vdots \\ \lambda_{s(\delta,t)}^t r_{s(\delta,t)}(i) \end{bmatrix}. \tag{28}$$

This allows us to determine the power t for embedding into $\mathbb{R}^d$ as follows. We pick $\delta \in (0, 1)$, for example, $\delta = 0.2$, and then define t so that t is the smallest integer such that

$$\left(\frac{|\lambda_d|}{|\lambda_1|}\right)^t \leq \delta \; \mathbb{R}ightarrow t = \mathsf{ceil}\left[\frac{\log(1/\delta)}{\log\left(|\lambda_1|/|\lambda_d|\right)}\right]. \tag{29}$$

Once the appropriate power for the desired dimension of embedding space (2 or 3) is found, we can define diffusion maps (abusing the term) to 2D or 3D diffusion spaces by

$$\Psi_t(x_i) = \begin{bmatrix} \lambda_1^t r_1(i) \\ \lambda_2^t r_2(i) \end{bmatrix} \quad \text{and} \quad \Psi_t(x_i) = \begin{bmatrix} \lambda_1^t r_1(i) \\ \lambda_2^t r_2(i) \\ \lambda_3^t r_3(i) \end{bmatrix}. \tag{30}$$

Now let us prove Proposition 1.

Proof. Let us redefine the diffusion kernel K as

$$K \to \left(\sum_{i=1}^{N} q_i\right)^{-1} K.$$

Then the stochastic matrix P with the new K can be decomposed as

$$P = \Pi^{-1}K.$$

P is similar to the symmetric matrix

$$A := \Pi^{1/2}P\Pi^{-1/2} = \Pi^{1/2}\Pi^{-1}K\Pi^{-1/2} = \Pi^{-1/2}K\Pi^{-1/2}.$$

Hence, the eigenvalues of A coincide with those of P. Let

$$A = \Phi\Lambda\Phi^\top = \sum_{k=0}^{n-1}\lambda_k\phi_k\phi_k^\top.$$

be an eigendecomposition of A where Φ is orthogonal, and the diagonal entries of Λ, the eigenvalues, are ordered in the decreasing order. Then the desired eigendecomposition of P can be obtained as follows:

$$P = \Pi^{-1/2}A\Pi^{1/2} = \Pi^{-1/2}\Phi\Lambda\Phi^\top\Pi^{1/2} =: R\Lambda L = \sum_{k=0}^{n-1}\lambda_k r_k l_k, \tag{31}$$

where $r_k := \Pi^{-1/2}\phi_k$, the columns of R, are the right eigenvectors of P, and $l_k := \phi_k^\top\Pi^{1/2}$, the rows of L, are the left eigenvectors of P. It can be readily verified that the left and right eigenvectors satisfy the following conjugacy relationships:

$$\sum_{m=1}^{n}\pi_m r_i(m)r_j(m) = r_i^\top\Pi r_j = \phi_i^\top\Pi^{-1/2}\Pi\Pi^{-1/2}\phi_j = \phi_i^\top\phi_j = \delta_{i,j}, \tag{32}$$

$$\sum_{m=1}^{n}\frac{l_i(m)l_j(m)}{\pi_m} = l_i\Pi^{-1}l_j^\top = \phi_i^\top\Pi^{1/2}\Pi^{-1}\Pi^{1/2}\phi = \phi_i^\top\phi_j = \delta_{i,j}. \tag{33}$$

Eq. (31) allows us to write entries of P^t as

$$p_{im}^t = \sum_{k=0}^{n-1}\lambda_k^t r_k(i)l_k(m). \tag{34}$$

Plugging p_{im}^t and p_{jm}^t into the definition of $D_t(i,j)$ (equation (23)), we get:

$$\begin{aligned}
D_t(x_i,x_j)^2 &= \sum_{m=1}^{n}\frac{1}{\pi_m}\left[\sum_{k=0}^{n-1}\lambda_k^t r_k(i)l_k(m) - \lambda_k^t r_k(j)l_k(m)\right]^2 \\
&= \sum_{m=1}^{n}\frac{1}{\pi_m}\sum_{k=0}^{n-1}\left[\lambda_k^t r_k(i)l_k(m) - \lambda_k^t r_k(j)l_k(m)\right]^2 \\
&+ \sum_{m=1}^{n}\sum_{k=0}^{n-1}\sum_{s\neq k}\frac{l_k(m)l_s(m)}{\pi_m}\lambda_k^t\lambda_s^t[r_k(i)-r_k(j)][r_s(i)-r_s(j)].
\end{aligned}$$

Let us show that the second term in this sum is zero. Rearranging the order of summation and using (33) we get

$$\sum_{k=0}^{n-1}\sum_{s\neq k}\lambda_k^t\lambda_s^t[r_k(i)-r_k(j)][r_s(i)-r_s(j)]\sum_{m=1}^{n}\frac{l_k(m)l_s(m)}{\pi_m}=0.$$

Returning to the first term, we calculate:

$$\begin{aligned}D_t(x_i,x_j)^2&=\sum_{m=1}^{n}\frac{1}{\pi_m}\sum_{k=0}^{n-1}\left[\lambda_k^t r_k(i)l_k(m)-\lambda_k^t r_k(j)l_k(m)\right]^2\\&=\sum_{m=1}^{n}\sum_{k=0}^{n-1}\frac{[l_k(m)]^2}{\pi_m}\lambda_k^{2t}\left[r_k(i)-r_k(j)\right]^2\\&=\sum_{k=0}^{n-1}\lambda_k^{2t}\left[r_k(i)-r_k(j)\right]^2\sum_{m=1}^{n}\frac{[l_k(m)]^2}{\pi_m}\\&=\sum_{n=0}^{n-1}\lambda_n^{2t}\left[r_n(i)-r_n(j)\right]^2.\end{aligned}$$

Finally, we take into account that since $r_0=[1,\dots,1]^\top$, $r_0(i)-r_0(j)=0$. Therefore,

$$D_t(x_i,x_j)=\sum_{k=1}^{n-1}\lambda_k^{2t}\left[r_k(i)-r_k(j)\right]^2$$

as desired. □

9.3. **Relation to Laplacian eigenmap.** As we have mentioned, the presented construction of the diffusion map is the most basic one and is very similar to the construction of Laplacian eigenmap [17]. Step 1 of Laplacian eigenmap is the construction of a graph with vertices at the data points which is done in one of the two following ways. Vertices x_i and x_j are connected by an edge

- if $\|x_i-x_j\|<\epsilon$, or
- if x_i is among k nearest neighbors of x_j or x_j is among k nearest neighbors of x_i.

Then, either way, the kernel matrix K is defined by

$$k_{ij}=\begin{cases}\exp\left(-\frac{\|x_i-x_j\|^2}{t}\right), & \|x_i-x_j\|<\epsilon,\\ 0, & \text{otherwise.}\end{cases},$$

where t is interpreted as time for the heat equation on the manifold occupied with the data. Note that $t=\infty$ corresponds to $k_{ij}=1$ if x_i and x_j are connected and zero otherwise, i.e., K is merely the adjacency matrix.

Next, we set up the matrix called the *graph Laplacian*:

$$L := Q - K, \quad \text{where} \quad Q = \mathsf{diag}\left\{\sum_j k_{1j}, \ldots, \sum_j k_{nj}\right\}. \tag{35}$$

Then the following generalized eigenvalue problem is solved:

$$LR = QRM, \quad M = \mathsf{diag}\{\mu_0, \mu_1, \ldots, \mu_{n-1}\}, \tag{36}$$

where

$$0 = \mu_0 \le \mu_1 \le \ldots \le \mu_{n-1}. \tag{37}$$

Note that the matrix of the right eigenvectors R coincides with that of P. The matrix M relates to Λ via

$$\Lambda = I - M. \tag{38}$$

Finally, the *Laplacian eigenmap* to $\mathbb{R}^m$, $m \le n$, is defined by

$$x_i \mapsto (r_1(i), \ldots, r_m(i)). \tag{39}$$

Exercise Show that the Laplacian eigenmap to $\mathbb{R}^m$ is the solution to the following optimization problem:

$$\sum_{i,j} k_{ij}\|y_i - y_j\|_2^2 = \mathsf{tr}\left(Y^\top L Y\right) \to \min \quad \text{subject to} \quad Y^\top Q Y = I, \quad Y^\top Q 1_{n\times 1} = 0. \tag{40}$$

Here, y_i's are columns of Y, and Y is $n \times m$.

Remark It is shown in [17] that LLE and Laplacian eigenmap are closely related. The minimization problem for LLE involves graph Laplacian squared.

9.4. **Illustrative examples.**

9.4.1. *Swiss Roll.* First we make an approximately uniform mesh of points on the Swiss Roll as shown in Fig. 6(a). The number of points is $n = 1060$. We set $\delta = 0.2$ and find the values for ϵ and t as described above:

$$\epsilon = 0.7717928, \quad t = 147.$$

The points are sorted and colored according to the approximate geodesic distance to the data point closest to the origin. The matrix P^t is displayed in Fig. 6(b). The absolute eigenvalues of P^t starting from $|\lambda_1|$ are shown in Fig. 6(c). The embedding to 3D is in Fig. 6(d). The Swiss Roll has been mostly unrolled.

Next, we repeat this experiment by adding noise to the data:

```
noisestd = 0.4;
X = X + noisestd*randn(size(X)); % perturb by Gaussian noise
```

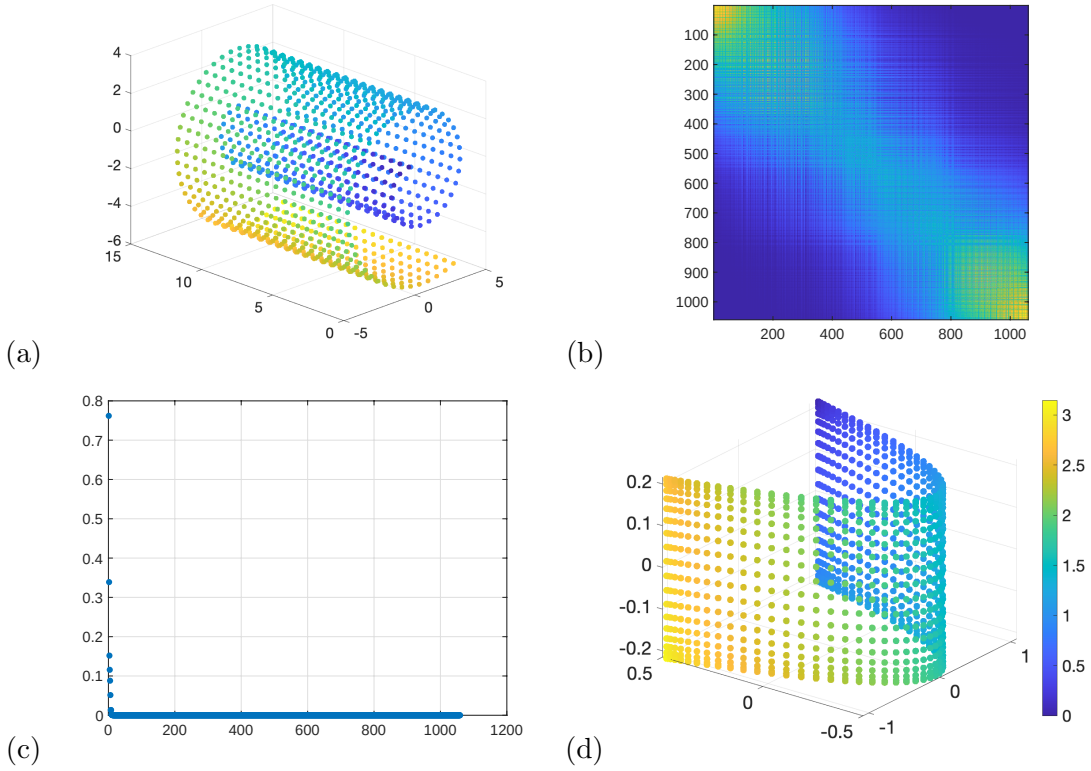

FIGURE 6. (a): The Swiss Roll dataset with points arranged into a quasi-uniform mesh. (b): The matrix P^t for $\epsilon = 0.7717928$, $t = 147$. (c): The absolute eigenvalues of the eigenvalues of P^t starting from $|\lambda_1|$. (d): Diffusion map to 3D.

Setting $\delta = 0.2$ as before, we find:

$$\epsilon = 0.6108029, \quad t = 300.$$

The results are shown in Fig. 7.

Finally, we take the same Swiss Roll data that we used for the isomap experiments with Gaussian noise of standard deviation 0.8. With $\delta = 0.2$, we found

$$\epsilon = 2.104531, \quad t = 1705.$$

The results are shown in Fig. 8.

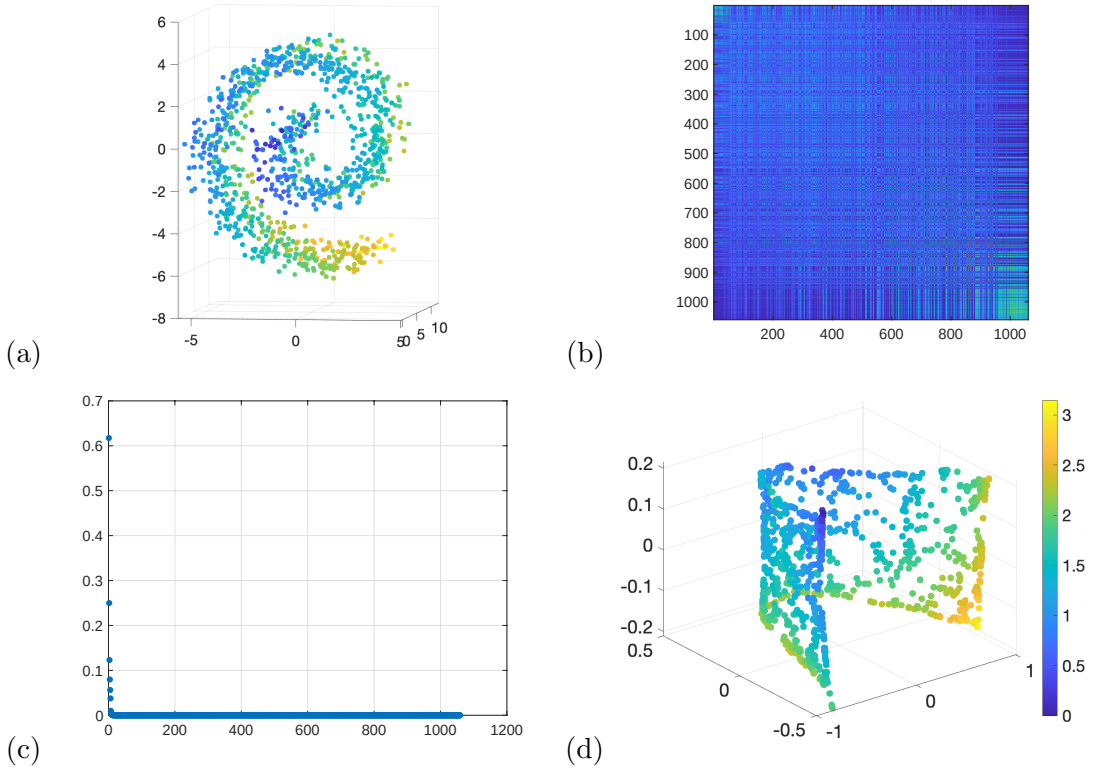

FIGURE 7. (a): The Swiss Roll dataset with points arranged into a quasi-uniform mesh and perturbed by Gaussian noise with standard deviation 0.4. (b): The matrix P^t for $\epsilon = 0.6108029$, $t = 300$. (c): The absolute eigenvalues of the eigenvalues of P^t starting from $|\lambda_1|$. (d): Diffusion map to 3D.

9.4.2. *Pacman.* Let us consider a data set consisting of 200 images depicting the Pacman. This example is similar to the one in this article[1]. Each image is 65×65 pixels either black (color = 0) or white (color = 255). The images differ from each other by the angle of rotation of the Pacman around the center of the image. The angles of rotation are

$$\alpha_i = \frac{2\pi i}{200}.$$

A sample of 20 such images is shown in Fig. 9(a). This dataset is naturally embedded into $\mathbb{R}^{65^2} = \mathbb{R}^{4225}$ space. Note that $D > N$ in this case. The PCA mapping into 3D applied to

[1]While this article offers a nice exposition, I do not recommend to rely on it as it contains a number of errors in important formulas. For example, Eqs. (6) and (7) contain errors, the comments following Eq. (9) are misleading, etc.

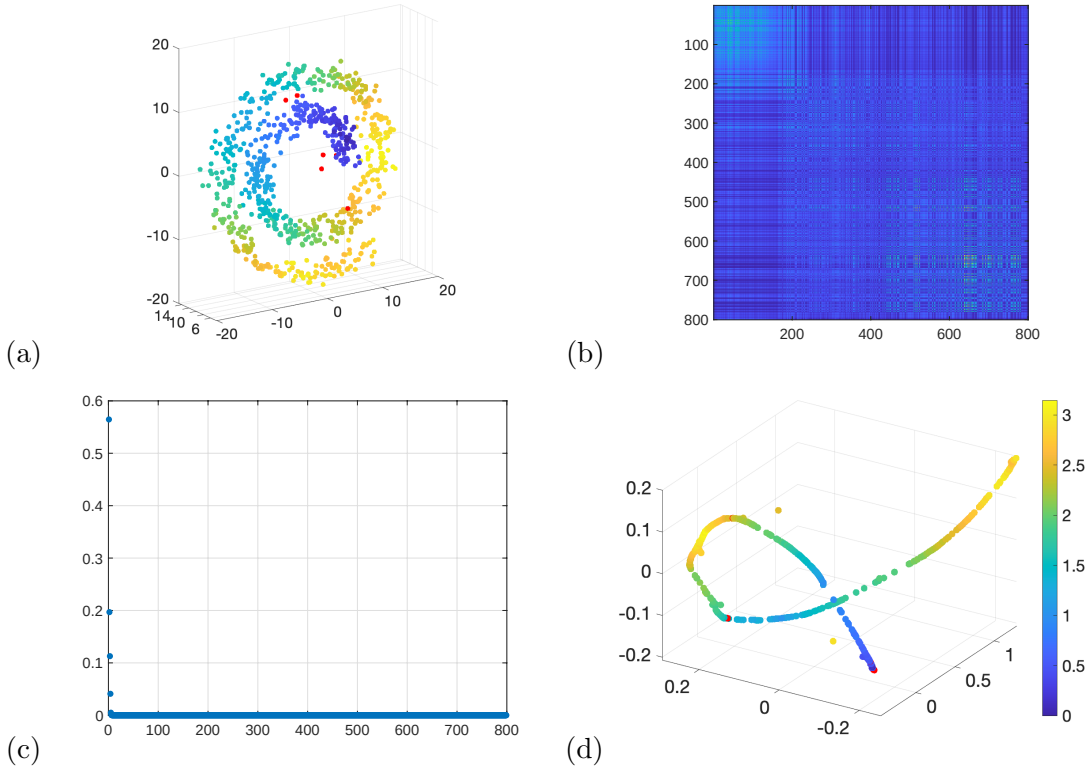

FIGURE 8. (a): The Swiss Roll dataset used for the experiments with isomap perturbed by Gaussian noise with standard deviation 0.8. (b): The matrix P^t for $\epsilon = 2.104531$, $t = 1705$. (c): The absolute eigenvalues of the eigenvalues of P^t starting from $|\lambda_1|$. (d): Diffusion map to 3D.

this dataset is shown in Fig. 9(b). The absolute eigenvalues and the embedding into 3D are shown in Figs. 9(c) and (d) respectively. Both the PCA and the diffusion map show that the set of images is well-approximated by a 1D manifold as we would expect.

9.4.3. *Cat-in-the-hat.* A similar example with a more complex image of the Cat-in-the-hat is shown in Fig. 10. Each image is 500×500. The double-loop formed by the mapped data is caused by the fact that the image rotated by π is closer to the original image than those rotated by an angle between $\pi/6$ and $5\pi/6$.

9.5. **The continuous counterpart of the diffusion map algorithm.** Reference for this section: [18] arXiv:2208.13772. The basic diffusion map algorithm presented in Section 9.2 leaves us with two questions. What is a good choice of the bandwidth parameter ϵ? What is a good choice of power t? The first question will be answered in Section 9.7 below. The second question will become irrelevant as we will renormalize the kernel and eliminate the

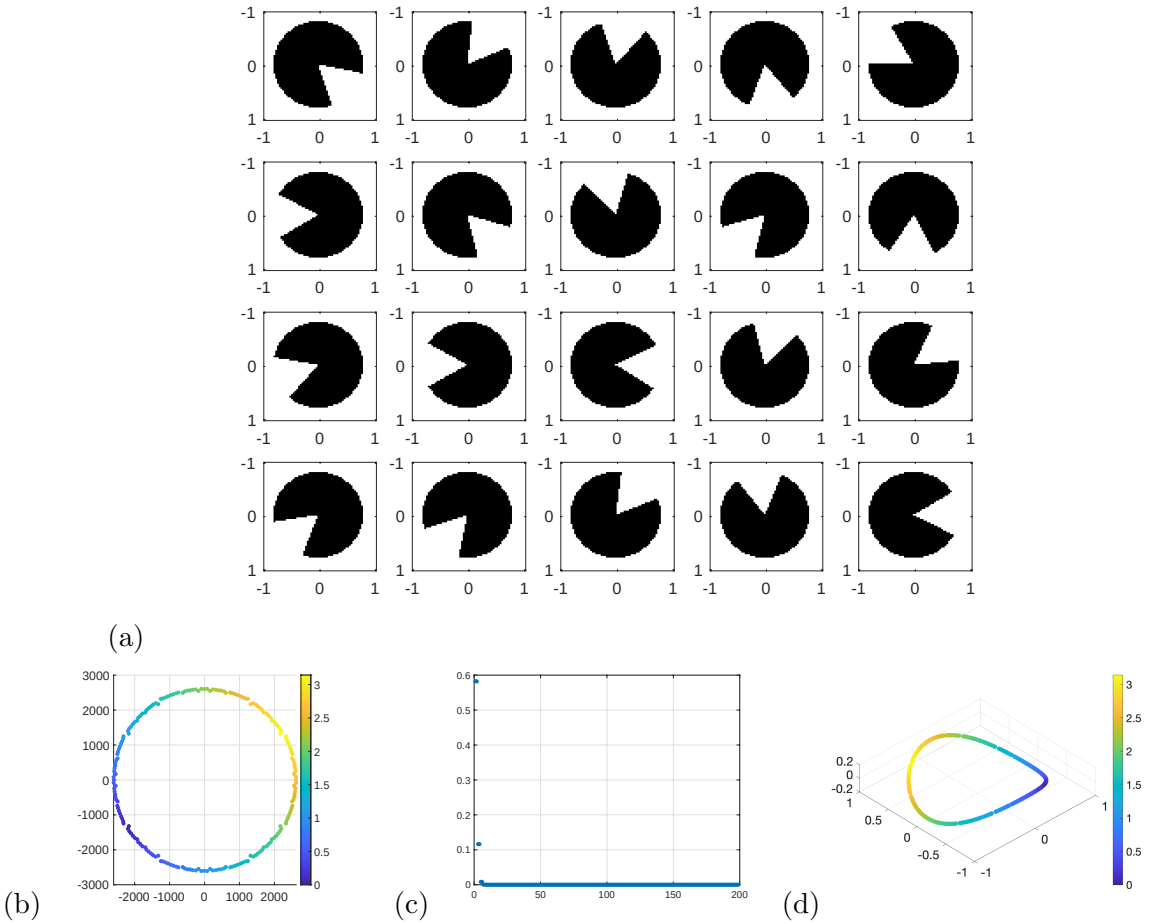

FIGURE 9. The dataset consists of 200 images of the Pacman rotated around the center of the image by angles $\alpha_i = 2\pi i/200$. (a): A sample of 20 data points. (b): The PCA mapping into 3D. (c): The absolute eigenvalues of the eigenvalues of P^t starting from $|\lambda_1|$. Here: $\delta = 0.5$, $\epsilon = 2335698$, $t = 187$. (d): Diffusion map to 3D.

need for taking power t altogether. To understand to answers to these questions, we will consider the continuous counterpart of the diffusion map algorithm also described in the paper by Coifman and Lafon (2006).

Why the diffusion map algorithm contains the word diffusion in its name? What is the underlying diffusion process? Looking at the basic construction we can think that the dynamics of the Markov chain with the constructed stochastic matrix P is, perhaps, a diffusion process. In fact, it is indeed a diffusion process discretized to a point cloud. To

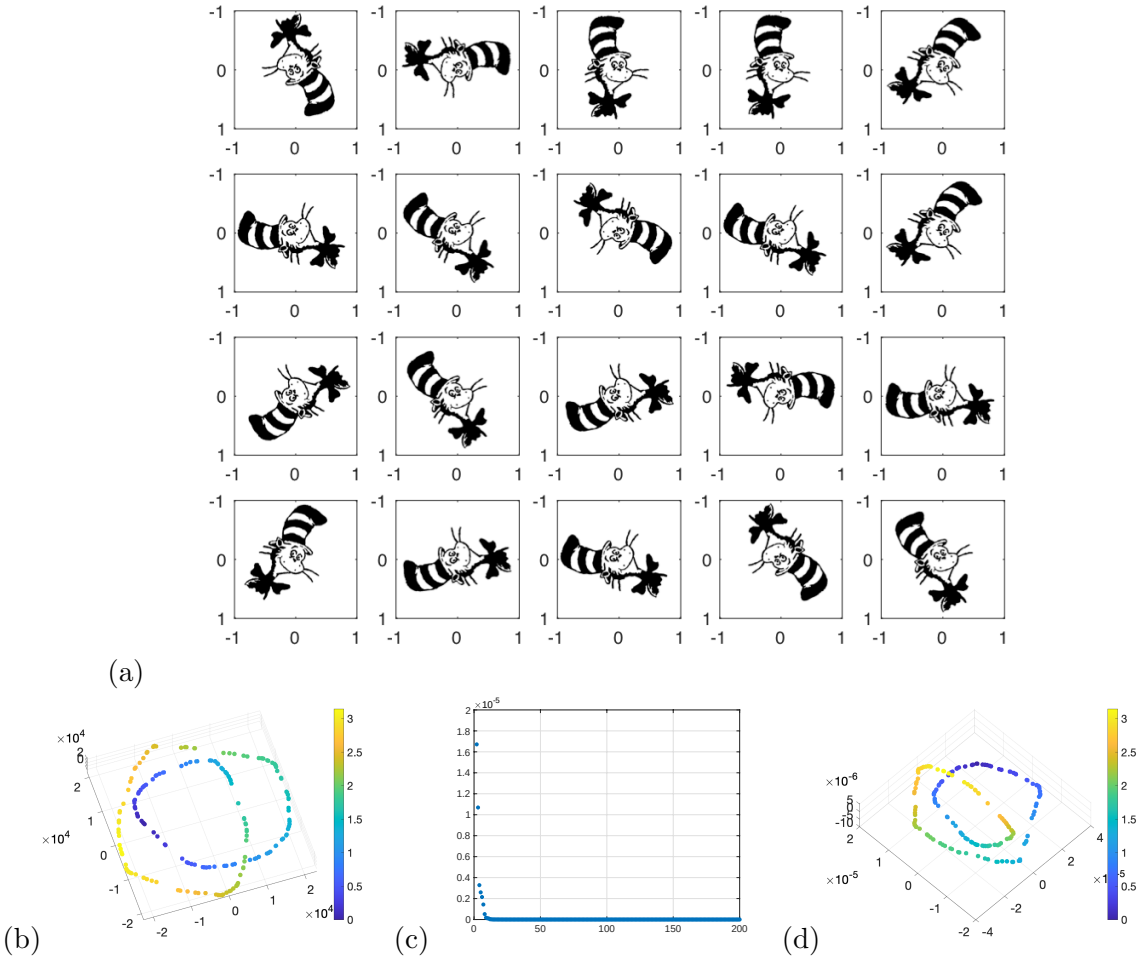

FIGURE 10. The dataset consists of 200 images of the Cat-in-the-hat rotated around the center of the image by angles uniformly distributed in $(0, 2\pi)$. (a): A sample of 20 data points. (b): The PCA mapping into 2D. (c): The absolute eigenvalues of the eigenvalues of P^t starting from $|\lambda_1|$. Here: $\delta = 0.2$, $\epsilon = 7.318159 \cdot 10^7$, $t = t$. (d): Diffusion map to 3D.

see it, let us start with a diffusion equation in $\mathbb{R}^d$:

$$u_t = \frac{1}{4}\Delta u, \quad x \in \mathbb{R}^d, \quad u(x,0) = f(x), \tag{41}$$

where $\Delta u = u_{x_1x_1} + u_{x_2x_2} + \ldots + u_{x_dx_d}$ is Laplace's operator applied to u. The solution to (41) at time $t = \epsilon$ given by

$$u(x,\epsilon) = \frac{1}{(\pi\epsilon)^{d/2}} \int_{\mathbb{R}^d} e^{-\frac{||x-x'||^2}{\epsilon}} f(x')dx'. \tag{42}$$

On the other hand, if ϵ is small, $u(x,\epsilon)$ can be found using a Taylor expansion:

$$u(x,\epsilon) = u(x,0) + \frac{\partial}{\partial t}u(x,0)\epsilon + O(\epsilon^2) = f(x) + \frac{\epsilon}{4}\Delta f(x) + O(\epsilon^2). \tag{43}$$

Matching (42) and (43), we obtain the following Taylor expansion for the integral operator with the Gaussian kernel $k_\epsilon(x,x') := \exp(-||x-x'||^2/\epsilon)$:

$$\frac{1}{(\pi\epsilon)^{d/2}} \int_{\mathbb{R}^d} e^{-\frac{||x-x'||^2}{\epsilon}} f(x')dx' = f(x) + \frac{\epsilon}{4}\Delta f(x) + O(\epsilon^2). \tag{44}$$

Now let us connect this integral operator with the matrix operators K and P constructed in Section 9.2. We observe that the normalizing factor $(\pi\epsilon)^{d/2}$ is obtained by integrating the kernel with respect to its second argument over $\mathbb{R}^d$:

$$(\pi\epsilon)^{d/2} = \int_{\mathbb{R}^d} e^{-\frac{||x-x'||^2}{\epsilon}} dx'. \tag{45}$$

Let X be an $n \times d$ data matrix whose rows $x_i^\top \in \mathbb{R}^d$ are the data points forming a point cloud. Let f be a smooth function with compact support. We discretize f to the point cloud $\{x_i\}_{i=1}^n$ by setting $f_i = f(x_i)$, $1 \le i \le n$, $[f] = [f_1, \ldots, f_n]^\top$, and compute $P[f]$. We have:

$$[P[f]]_i = \frac{[P[f]]_i}{[K1_{n\times 1}]_i} = \frac{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}} f(x_j)}{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}}}.$$

Taking limit as $n \to \infty$ we get:

$$\lim_{n\to\infty} \frac{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}} f(x_j)}{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}}} = \frac{\int_{\mathbb{R}^d} e^{-\frac{||x-x'||^2}{\epsilon}} f(x')\rho(x)dx'}{\int_{\mathbb{R}^d} e^{-\frac{||x-x'||^2}{\epsilon}} \rho(x)dx'},$$

where $\rho(x)$ is the probability density function that the point cloud is sampled from. Using the Taylor expansion (44) we calculate:

$$\lim_{n\to\infty} \frac{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}} f(x_j)}{\sum_j e^{-\frac{||x_i-x_j||^2}{\epsilon}}} = \frac{\int_{\mathbb{R}^d} e^{-\frac{||x_i-x'||^2}{\epsilon}} f(x')\rho(x)dx'}{\int_{\mathbb{R}^d} e^{-\frac{||x_i-x'||^2}{\epsilon}} \rho(x)dx'} \tag{46}$$

$$= \frac{(f\rho)(x_i) + \frac{\epsilon}{4}\Delta(f\rho)(x_i) + O(\epsilon^2)}{\rho(x_i) + \frac{\epsilon}{4}\Delta\rho(x_i) + O(\epsilon^2)}$$

$$= f(x_i) + \frac{\epsilon}{4}\left[\Delta f(x_i) + 2\nabla f(x_i) \cdot \frac{\nabla\rho(x_i)}{\rho(x_i)}\right] + O(\epsilon^2). \tag{47}$$

To obtain the last equality we used

$$\frac{1}{\rho + \frac{\epsilon}{4}\Delta\rho(x) + O(\epsilon^2)} = \frac{1}{\rho}\left(1 - \frac{\epsilon}{4}\frac{\Delta\rho}{\rho} + O(\epsilon^2)\right).$$

Therefore, subtracting $[f]$ from $P[f]$ and dividing the result by ϵ we obtain a point-wise approximation to the action of the differential operator

$$\frac{1}{4}\left[\Delta + 2\frac{\nabla\rho\cdot\nabla}{\rho}\right] \equiv \frac{1}{4}\left[\Delta + \nabla\log\rho^2\cdot\nabla\right] \tag{48}$$

on the function f:

$$\mathcal{L}_\epsilon f(x_i) := \lim_{\epsilon\to 0}\lim_{n\to\infty}\frac{[P[f]]_i - f_i}{\epsilon} = \frac{1}{4}\left[\Delta f(x_i) + \nabla\log\rho^2\cdot\nabla f(x_i)\right] \tag{49}$$

The operator $\frac{1}{\epsilon}(P - I)$ in the left-hand side of (49) is the generator of the Markov chain constructed in Section 9.2 for the discrete time steps of size ϵ. As we see, the generator approximates not the Laplacian but the differential operator (48). If the sampling density ρ were uniform in some region Ω, this operator would be proportional to Laplace's operator in Ω.

9.6. **Removing the effect of nonuniform sampling.** Usually, the sampling density of data points is nonuniform. In this case, it is advantageous to modulate the effect of nonuniform density by the right renormalization of the kernel function originally developed by Coifman and Lafon [10] and then simplified in later works [19, 20]. We define a family of right-renormalized kernels by

$$k_{\epsilon,\alpha}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\epsilon}}\rho_\epsilon^{-\alpha}(x_j), \tag{50}$$

where $\rho_\epsilon(x')$ is the estimate for the sampling density at x'. Note that typically the sampling density is not known but can be estimated using the fact that the Gaussian kernel with a proper normalization approximates the Dirac δ-function:

$$(\pi\epsilon)^{-d/2}e^{-\frac{\|x - x'\|}{\epsilon}} \approx \delta(x - x').$$

Indeed, it is easy to check using Taylor expansion in ϵ that

$$(\pi\epsilon)^{-d/2}\int_{\mathbb{R}^d} e^{-\frac{\|x - x'\|^2}{\epsilon}}\rho(x')dx' = \rho(x) + O(\epsilon). \tag{51}$$

Motivated by this, we define the following density estimate:

$$\rho_\epsilon(x_i) = (\pi\epsilon)^{-d/2}\frac{1}{n}\sum_j e^{-\frac{\|x_i - x_j\|^2}{\epsilon}}. \tag{52}$$

If the kernel $e^{-\frac{||x - x_j||^2}{\epsilon}}$ is replaced with the right-normalized kernel

$$e^{-\frac{||x - x_j||^2}{\epsilon}}\rho_\epsilon^{-\alpha}(x_j), \tag{53}$$

a calculation of a limit similar to the one in the left-hand side of (46) results in the following family of differential operators

$$\mathcal{L}_{\epsilon,\alpha} := \frac{1}{4}\left[\Delta f + \nabla f \cdot \nabla \left[\log \rho^{2(1-\alpha)}\right]\right], \tag{54}$$

For $\alpha = 1$, the resulting operator is the Laplacian. Another case of interest is $\alpha = 0.5$. In this case, the generator $\mathcal{L}_{\epsilon,0.5}$ is the generator for the overdamped Langevin dynamics.

Let us summarize the diffusion map algorithm with $\alpha = 1$.

- **Step 1.** Set a rotation-invariant kernel

$$k_\epsilon(x,y) = \exp\left[-\|x-y\|^2/\epsilon\right].$$

To make the kernel matrix sparse, define

$$[K_\epsilon]_{ij} = \begin{cases} \exp\left[-\|x_i - x_j\|^2/\epsilon\right], & \|x_i - x_j\| < 3\sqrt{0.5\epsilon}, \\ 0, & \text{otherwise.} \end{cases}$$

- **Step 2.** Calculate row sums $q_\epsilon(x) = \sum_y k_\epsilon(x,y)$ and form the new kernel

$$k_\epsilon^{(1)}(x_i,x_j) = \frac{k_\epsilon(x_i,x_j)}{q_\epsilon(x_j)}, \quad \text{or} \quad K = KQ^{-1}. \tag{55}$$

- **Step 3.** Calculate row sums

$$d_\epsilon^{(1)}(x_i) = \sum_j k_\epsilon^{(1)}(x_i,x_j)$$

and define the stochastic matrix

$$P_{\epsilon,1} = \left[D_\epsilon^{(1)}\right]^{-1} K_\epsilon^{(1)}, \tag{56}$$

where

$$D_\epsilon^{(1)} = \mathsf{diag}\left\{d_\epsilon^{(1)}(x_1),\ldots,d_\epsilon^{(1)}(x_n)\right\}, \quad K_\epsilon^{(1)} = (k_\epsilon^{(1)}(x_i,x_j))_{i,j=1}^n.$$

The rest of the construction is the same as in Section 9.2 except for the formula for the embeddings to 2D or 3D are

$$\Psi(x_i) = \begin{bmatrix} r_1(i) \\ r_2(i) \end{bmatrix}, \quad \Psi(x_i) = \begin{bmatrix} r_1(i) \\ r_2(i) \\ r_3(i) \end{bmatrix}. \tag{57}$$

9.7. **Choosing ϵ.** Reference for this section: [18] arXiv:2208.13772. In practice, the limit $\epsilon \to 0$ cannot be taken for a finite dataset. Instead, one generally tries to choose ϵ as small as possible without making the corresponding generator matrix $L_{\epsilon,\mu}$ reducible. Many heuristics exist for choosing the scaling parameter ϵ in diffusion maps, relating back to bandwidth selection in kernel density estimation [13]. Here, we present the method of Berry, Harlim and Giannakis [14, 15, 16, 21] which we refer to as the "Ksum test".

The idea for the heuristic is to find the range of ϵ where the asymptotic results of diffusion maps hold true for the given dataset. We find this range by analyzing the double sum

$$S(\epsilon) := \frac{1}{N^2}\sum_{i=1}^{n}\sum_{j=1}^{n}[K_\epsilon]_{ij}$$

over a range of ϵ values. Here, $[K_\epsilon]_{ij} = k_\epsilon(x_i, x_j)$ where $k_\epsilon(x, x') = \exp(-\|x - x'\|^2/\epsilon)$ is the Gaussian kernel. We assume that the point cloud is located in a finite region $\Omega \subset \mathbb{R}^d$. For large n, the intermediate asymptotic for $S(\epsilon)$ is [22, 15]

$$S(\epsilon) \approx \int_\Omega dx \int_\Omega dx' k_\epsilon(x, x') \approx \pi^{d/2}\epsilon^{d/2}\mathsf{vol}(\Omega) \equiv C\epsilon^{d/2}$$

where C is a constant independent of ϵ. Hence,

$$\log S(\epsilon) \approx \frac{d}{2}\log\epsilon + \log C, \tag{58}$$

i.e., $\log S$ is a linear function of $\log\epsilon$ if ϵ is not too large and not too small.

On the other hand, if ϵ is large, $[K_\epsilon]_{ij} \approx 1$ for all i, j, and hence $S(\epsilon) \to 1$ as $\epsilon \to \infty$. For small ϵ, $[K_\epsilon]_{ij} \approx 0$ for all i, j, $i \neq j$, and $[K_\epsilon]_{ii} = 1$. Therefore, $S(\epsilon) \to N^{-1}$ as $\epsilon \to 0$. Therefore, if we plot $\log\epsilon$ against $\log S(\epsilon)$, we expect to see a linear region of slope approximately $\frac{d}{2}$, where d is the dimension of the dataset.This region demarcates the range of suitable values of ϵ [14, 15, 16, 21]. On the other hand, the slope of this graph should tend to zero as $\epsilon \to 0$ and as $\epsilon \to \infty$.

In particular, we expect to have $\frac{\partial \log S(\epsilon)}{\partial \log \epsilon} \approx \frac{d}{2}$ where the slope is maximal. For a practical calculation, it is useful to note that the slope is given by

$$\frac{\partial \log S(\epsilon)}{\partial \log \epsilon} = -\frac{\sum_{i,j=1}^{N}[K_\epsilon]_{ij}\log[K_\epsilon]_{ij}}{\sum_{i,j=1}^{N}[K_\epsilon]_{ij}}. \tag{59}$$

References

[1] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *Journal of Machine Learning Research*, vol. 16, pp. 2859–2900, 2015.

[2] I. Jolliffe, *Principal component analysis*. Springer-Verlag New York, 1986.

[3] I. Borg and P. J. F. Groenen, *Modern multidimensional scaling: theory and applications*. Springer, 2005.

[4] V. de Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," tech. rep., Stanford University, 2004.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, Second Edition*. Wiley and Sons, 2001.

[6] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[7] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[8] L. van der Maarten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[9] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," *Neural Information Processing Systems (NIPS'02)*, vol. 15, pp. 357–864, 2003.

[10] R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 5–30, 2006.
[11] A. L. Ferguson, A. Z. Panagiotopoulosa, P. G. Debenedettia, and K. I. G., "Systematic determination of order parameters for chain dynamics using diffusion maps," *PNAS*, vol. 107, no. 31, pp. 13597–13602, 2010.
[12] S. B. Kim, C. J. Dsilva, I. G. Kevrekidis, and P. G. Debenedetti, "Systematic characterization of protein folding pathways using diffusion maps: Application to trp-cage miniprotein," *J. Chem. Phys*, vol. 142, p. 085101, 2019.
[13] O. Lindenbaum, M. Salhov, A. Yeredor, and A. Averbuch, "Gaussian bandwidth selection for manifold learning and classification," *Data mining and knowledge discovery*, vol. 34, no. 6, pp. 1676–1712, 2020.
[14] T. Berry, D. Giannakis, and J. Harlim, "Nonparametric forecasting of low-dimensional dynamical systems," *Physical Review E*, vol. 91, no. 3, p. 032915, 2015.
[15] T. Berry and J. Harlim, "Variable bandwidth diffusion kernels," *Applied and Computational Harmonic Analysis*, vol. 40, no. 1, pp. 68–96, 2016.
[16] D. Giannakis, "Data-driven spectral decomposition and forecasting of ergodic dynamical systems," *Applied and Computational Harmonic Analysis*, vol. 47, no. 2, pp. 338–396, 2019.
[17] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput*, vol. 13, pp. 1373–1397, 2003.
[18] A. L. Evans, M. K. Cameron, and P. Tiwary, "Computing committors via mahalanobis diffusion maps with enhanced sampling data," *Journal of Chemical Physics*, 2022. arXiv:2208.13772.
[19] R. Banisch, Z. Trstanova, A. Bittracher, S. Klus, and P. Koltai, "Diffusion maps tailored to arbitrary non-degenerate itô processes," *Applied and Computational Harmonic Analysis*, vol. 48, no. 1, pp. 242–265, 2020.
[20] Z. Trstanova, B. Leimkuhler, and T. Lelièvre, "Local and global perspectives on diffusion maps in the analysis of molecular systems," *Proceedings of the Royal Society A*, vol. 476, no. 2233, p. 20190036, 2020.
[21] A. D. Davis and D. Giannakis, "Graph-theoretic algorithms for kolmogorov operators: Approximating solutions and their gradients in elliptic and parabolic problems on manifolds," *arXiv preprint arXiv:2104.15124*, 2021.
[22] R. R. Coifman, I. G. Kevrekidis, S. Lafon, M. Maggioni, and B. Nadler, "Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems," *Multiscale Modeling & Simulation*, vol. 7, no. 2, pp. 842–864, 2008.