Due: 19 Nov 2020

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

$$\min \sum_{i,j} k_{ij} \|y_i - y_j\|_2^2 \qquad \text{subject to} \qquad Y^\intercal Q Y = I, \qquad Y^\intercal Q 1_{m\times 1} = 0.$$

Here, the y_i are rows of $Y \in \mathbb{R}^{n\times m}$ and the rest of the notation is as in the lecture notes. Also show that $\sum_{i,j} k_{ij}\|y_i - y_j\|_2^2 = \operatorname{tr}(Y^\intercal L Y)$ where L is the graph Laplacian, $L = Q - K$.

Solution. We begin by showing that $\sum_{i,j} k_{ij}\|y_i - y_j\|_2^2 = \operatorname{tr}(Y^\intercal L Y)$ using the fact that K is symmetric and $q_i = \sum_j k_{ij}$.

$$\begin{aligned}
\sum_{i,j} k_{ij}\|y_i - y_j\|_2^2 &= \sum_{i,j} k_{ij}(\|y_i\|^2 + \|y_j\|^2 - 2y_i^\intercal y_j) \\
&= \sum_{i,j} k_{ij}\|y_i\|^2 + \sum_{i,j} k_{ij}\|y_j\|^2 - 2\sum_{i,j} k_{ij} y_i^\intercal y_j \\
&= \sum_i q_i \|y_i\|^2 + \sum_j q_j \|y_j\|^2 - 2\sum_{i,j} k_{ij} y_i^\intercal y_j \\
&= 2\operatorname{tr}(Y^\intercal Q Y) - 2\sum_{i,j} k_{ij} y_i^\intercal y_j.
\end{aligned}$$

Next, we show that $\operatorname{tr}(Y^\intercal K Y) = \sum_{i,j} k_{ij} y_i^\intercal y_j$.

$$\begin{aligned}
\operatorname{tr}(Y^\intercal K Y) = \operatorname{tr}(K Y Y^\intercal) &= \operatorname{tr}\left(K \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \begin{bmatrix} y_1^\intercal & \cdots & y_n^\intercal \end{bmatrix} \right) = \operatorname{tr}\left(K \begin{bmatrix} y_1 y_1^\intercal & \cdots & y_1 y_n^\intercal \\ \vdots & \ddots & \vdots \\ y_n y_1^\intercal & \cdots & y_n y_n^\intercal \end{bmatrix} \right) \\
&= \operatorname{tr}\left(K \begin{bmatrix} y_1 y_1^\intercal & \cdots & y_n^\intercal y_1 \\ \vdots & \ddots & \vdots \\ y_1^\intercal y_n & \cdots & y_n^\intercal y_n \end{bmatrix} \right) = \operatorname{tr}\left(\begin{bmatrix} \sum_i k_{1i} y_i^\intercal y_1 & \cdots & \sum_i k_{1i} y_i^\intercal y_n \\ \vdots & \ddots & \vdots \\ \sum_i k_{ni} y_i^\intercal y_1 & \cdots & \sum_i k_{ni} y_i^\intercal y_n \end{bmatrix} \right) \\
&= \sum_j \sum_i k_{ji} y_i^\intercal y_j = \sum_{i,j} k_{ij} y_i^\intercal y_j
\end{aligned}$$

which is exactly the same term as in the earlier calculation. So we see that

$$\sum_{i,j} k_{ij}\|y_i - y_j\|_2^2 = 2\operatorname{tr}(Y^\intercal Q Y) - 2\operatorname{tr}(Y^\intercal K Y) = 2\operatorname{tr}(Y^\intercal L Y)$$

as desired (up to normalization by a factor of $1/2$).

We next have to prove that this trace is minimized by the Laplacian eigenmap to $\mathbb{R}^m$. Begin with the minimization problem $\min_Y \operatorname{tr}(Y^\intercal L Y)$ and impose the constraint $Y^\intercal Q Y = I$. This constraint removes a scale factor in the embedding—by forcing $Y^\intercal Q Y = I$, we're merely normalizing the embedded vectors y_i under the inner product defined by Q. Next, we find the Lagrangian of this constrained optimization problem:

$$\mathbb{L} = \operatorname{tr}(Y^\intercal L Y) - \operatorname{tr}(\Lambda^\intercal (Y^\intercal Q Y - I))$$

where Λ is a diagonal matrix of Lagrange multipliers. Taking the derivative and setting it to zero gives

$$\frac{\partial \mathbb{L}}{\partial Y} = 2LY - 2QY\Lambda = 0 \implies LY = QY\Lambda$$

which is exactly the generalized eigenvalue problem that we solve when we find the Laplacian eigenmap.

To get the last constraint, we notice that an easy to see eigenpair is $(0, 1_{m\times 1})$:

$$L 1_{m\times 1} = (Q - K) 1_{m\times 1} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} - \begin{bmatrix} \sum_j k_{1j} \\ \vdots \\ \sum_j k_{nj} \end{bmatrix} = 0.$$

From this, we impose the constraint $Y^\top Q 1_{m\times 1} = 0$, meaning that all of the other eigenvectors must be orthogonal to $1_{m\times 1}$ under the inner product defined by Q. This allows us to eliminate the trivial eigenpair $(0, 1_{m\times 1})$ and makes it so the embedding works.

2. **Dimensional reduction comparison.** The goal of this problem is to practice and compare various methods for dimensional reduction. Use the following methods:

- PCA
- Isomap
- LLE
- t-SNE
- Diffusion maps.

Diffusion maps should be programmed from scratch, but the rest can be programmed using library functions. If you use library functions, specify their source, read their descriptions, and be ready to adjust their parameters. There are three datasets:

- an S-curve generated by a Matlab function
- an S-curve perturbed by Gaussian noise—try various intensities and push the methods' limits
- the "emoji" dataset generated by a Matlab function—picking a good ϵ for diffusion map might be tricky since the nearest neighbor distances are very nonuniform. You should be able to get a nice 2D surface in 3D with the right ϵ. Use either $\alpha = 0$ or $\alpha = 1$ (up to you).

Submit a report on the performance of these methods on each dataset.

Solution. I generated the datasets in Matlab and saved them to files, then read and processed them in Python. I used functions from the `scikit-learn` library to do PCA, Isomap, LLE, and t-SNE. Before I began, I compiled a brief list of the parameters I could change in these functions that would significantly affect the performance of the methods:

- PCA (none)
- Isomap
  - `n_neighbors` = number of neighbors to consider when constructing graph (default 5)
- LLE
  - `n_neighbors` = number of neighbors to consider when constructing graph (default 5)
- t-SNE
  - `perplexity` = continuous measure of nearest neighbors (default 30, suggested $[5, 50]$)
  - `early_exaggeration` = controls how close together original clusters become in the embedded space (default 12)
  - `learning_rate` = step size for optimization problem. "If the learning rate is too high, the data may look like a 'ball' with any point approximately equidistant from its nearest neighbours. If the learning rate is too low, most points may look compressed in a dense cloud with few outliers." (default 200, suggested $[10, 1000]$)
- Diffusion map (custom code)
  - `epsilon` = size of ball within which to look for neighbors. I followed Lafon and Lee ("Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization"; 2006) and set $\epsilon = \frac{1}{n}\sum_{i=1}^{n} \min_{x_j \neq x_i} \|x_i - x_j\|^2$ so that it is adaptively adjusted based on the inputted data. However, there is still the option in my code to enter in a custom epsilon value.
  - `delta` = accuracy of embedding; determines the power for the stochastic matrix (default 0.2)
  - `alpha` = parameter for dealing with nonuniform data (default 0, suggested 0, 1/2, or 1)

Dataset 1: Regular S-curve. As noted in class, PCA did not unfold this dataset, but all the other methods provided decent embeddings into 3D. Isomap is perhaps the worst-performing here—the curves are jittery and not smooth as in all the other methods. Also as predicted, LLE provides a nice 2D manifold in $\mathbb{R}^3$. Since t-SNE is highly susceptible to initialization and parameters, I never got the same surface twice from it when I ran it, but they were all smooth and two-dimensional. Finally, diffusion map gives another smooth 2D manifold. Overall, all the methods worked satisfactorily on this dataset. See Figure 1 for plots.

Dataset 2: Noisy S-curves. Once again, PCA did not unfold the dataset, but instead simply replicated the original noisy data. **UNFINISHED**

Dataset 3: Emojis. UNFINISHED

Notes: PCA won't unfold dataset 1, all other methods should work. LLE should be good in 3d but not 2d. Remember t-SNE is a heuristic technique and there isn't excellent mathematical theory underlying it. It's made more for clustering, so it may separate faces into clusters. For emoji data, there are two points that will be quite far away from the others—they will screw up a bunch of the techniques, except diffusion maps (with good parameters you can eliminate the issue). Feel free to remove them—most smiley not smeared and smiley face with second smallest amount of smearing. Then the other techniques will also work better.

At what point will noise destroy all the methods?

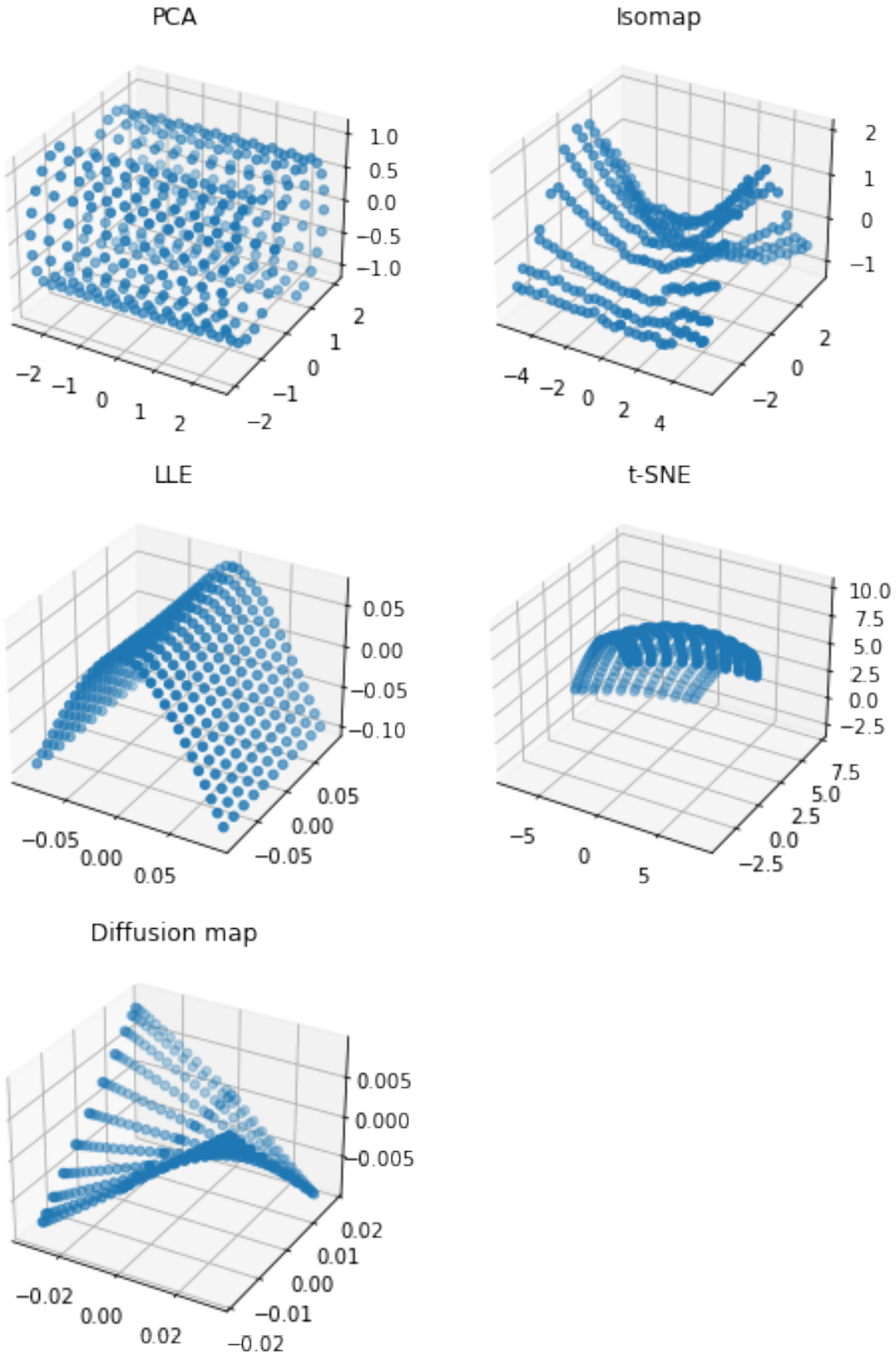

Figure 1: Application of each method on the regular S-curve.