# Geometric Data Analysis HW 2

Gilad Turok, gt2453
gt2453@columbia.edu

March 14, 2023

## 1 LLE

### 1.1 Solving for Reconstruction Weights

Consider that the optimal weights $w$ for reconstructing $x_i$ from an affine combination of its neighbors $x_j$. This is given by the solution to $w_i$ that solves $G_i w_i = \mathbf{1}$ for Gram matrix $G$ and normalized weight vector $w_i$.

*Proof.* Consider the (squared) cost function that quantifies how accurately we reconstruct data point $x_i$ from an affine combination ($\sum_j w_{ij} = 1$) of its $k$ nearest neighbors:

$$\begin{aligned}\mathcal{E}_i &= ||x_i - \sum_{j=1}^{k} w_{ij} x_j||^2 \\ &||(w_{i1} \dots w_{ik})x_i - \sum_j w_{ij} x_j||^2 \\ &= ||\sum_j w_{ij}(x_i - x_j)||^2\end{aligned}$$

Note that if data points $x_1 \dots x_n$ are scaled, rotated, or translated, the weights $w_i$ are the same. Now let $z$ be the difference between $x_i$ and its neighbors $x_j$:

$$z = \begin{bmatrix} x_i - x_1 \\ \vdots \\ x_i - x_k \end{bmatrix}$$

Rewrite our cost function:

$$\mathcal{E}_i = ||\sum_j w_{ij} z||^2 = (w_i^T z)(w_i^T z)^T = w_i^T (zz^T) w_i$$

Now define the gram matrix such that $G_{jk} = (x_i - x_j) \cdot (x_i - x_k)$:

$$G_i = \begin{bmatrix} (x_i - x_1) \cdot (x_i - x_1) & \dots & (x_i - x_1) \cdot (x_i - x_k) \\ \vdots & \ddots & \vdots \\ (x_i - x_k) \cdot (x_i - x_1) & \dots & (x_i - x_k) \cdot (x_i - x_k) \end{bmatrix}$$

Rewrite cost function again:

$$\mathcal{E}_i = w_i^T G_i w_i$$

Now we have a constrained optimization problem, minimizing the cost function subject to the weights summing to 1 – mathematically written as $\mathbf{1}^T w_i = 1$. Consider the Lagrange multiplier problem:

$$\begin{aligned} \mathcal{L}(w_i, \lambda) &= w_i^T G_i w_i - \lambda(\mathbf{1}^T w_i - 1) \\ \frac{\partial \mathcal{L}}{\partial w_i} &= 2G_i w_i - \lambda \mathbf{1} \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= \mathbf{1}^T w_i - 1 \end{aligned}$$

Thus we find the optimal weights must solve $2G_i w_i = \lambda \mathbf{1}$ with a $\lambda$ chosen such that the weights sum up to one. To do so, we can simply solve $G_i w_i = \mathbf{1}$ then normalize $w_i$. □

## 1.2 Solving for Lower Dimensional Embedding

Want to show that for fixed weight matrix $W = [w_1 \dots w_n]$, the ideal lower dimensional embeddings $y_i \in \mathbb{R}^m$ are given by the smallest $m$ eigenvectors of $M = (I - W)^T (I - W)$ after discarding the smallest eigenvector.

*Proof.* Seek to minimize the cost of lower dimensional dimensional data points $Y = [y_1 \dots y_n]$ given a fixed weight matrix $W = [w_1 \dots w_n]$:

$$\begin{aligned} \phi(Y) &= \sum_i ||y_i - \sum_j w_{ij} y_j||^2 \\ &= Y^T (I - W)^T (I - W) Y \\ &= Y^T M Y \qquad \text{for } M = (I - W)^T (I - W) \end{aligned}$$

Consider two constraints for lower dimensional embeddings $y_i$: ensure $y_i$ has dimension/rank $m$ and embedding is centered at the origin. Mathematically, these constraints are:

$$\frac{1}{n}Y^TY = 1 \qquad\qquad \sum Y_i = 0$$

Our cost function is (trivially) translationally equivariant with respect to $y_i$ so the second constraint is taken care of. To satisfy the first constraint, consider the Lagrange multiplier:

$$\begin{aligned}
\mathcal{L}(Y,\mu) &= Y^TMY - \mu(\frac{1}{n}Y^TY - 1) \\
\frac{\partial \mathcal{L}}{\partial Y} &= 2MY - \frac{2\mu}{n}Y = 0 \\
\frac{\partial \mathcal{L}}{\partial \mu} &= Y^TMY
\end{aligned}$$

From the second equation, we observe that $MY = \frac{\mu}{n}Y$ which is satisfied by setting $y_i$ equal to the eigenvectors of $M$. Therefore, to minimize $Y^TMY$, we set $Y$ to the $m$ smallest eigenvectors of matrix $M$.

However, note that the smallest eigenvector-eigenvalue pair of $(\mathbf{1}, 0)$ is trivial. Thus we ultimately minimize the cost by setting $Y$ to the smallest $m$ eigenvalues after discarding the very smallest eigenvector. □

## 2 Non-Manifold Spaces

An example of a non-manifold space is two unit circles, one centered at $(-1,0)$ and the other at $(1,0)$, that touch at a single point $(0,0)$:

$$\begin{aligned}
(x-1)^2 + y^2 &= 1 \\
(x+1)^2 + y^2 &= 1
\end{aligned}$$

This space is not a manifold because at the overlapping point $(0,0)$, a propert chart cannot be constructed. It's neighborhood looks like a rotated '+' sign which is not homeomorphic to a line segment in $\mathbb{R}^1$.

## 3 Chart Mapping to Zero

Show that given any point $m \in \mathcal{M}$, where $\mathcal{M}$ is a topoligical manifold, we can choose a chart $\theta : U \to \mathbb{R}^n$ such that $\theta(m) = 0$.

*Proof.* Consider all neighborhoods that contain $m$ in its domain i.e. $m \in U_1, U_2, \ldots$. By definition of a manifold, for each neighborhood $U_i$, there exists some corresponding chart $\theta_i : U_i \to \mathbb{R}^n$ where $m$ is mapped to some arbitrary location $p \in \mathbb{R}^n$.

To instead make $\theta_i(m) = 0$, we can simply translate the chart by $-p$. Thus we can construct entirely new charts:

$$\theta_i'(x) = \theta_i(x) - p = \theta_i(x) - \theta_i(m)$$

which ensures $\theta_i(m) = 0$ always.

□

## 4 Coordinate Charts and Transition Functions for Circles

We will explicitly construct coordinate charts and transition functions for a circle defined by the angle and $x, y$ projections respectively. To do so, consider $S$, the unit circle $x^2 + y^2 = 1$ defined for $[0, 2\pi]$.

### 4.1 Angle Construction

Let $\alpha$ be some angle less than $\pi$ and let $\phi$ be the angle between a point $(x, y) \in S$ and the $x$ axis. Now divide the circle into two coordinate patches:

$$\begin{aligned} U_1 &= \{(x, y) \in S | -\alpha \leq \phi \leq \pi + \alpha\} \\ U_2 &= \{(x, y) \in S | \pi \leq \phi \leq 2\pi\} \end{aligned}$$

Now define two corresponding charts $\theta : U \to \mathbb{R}$. I do so based on the arctan2 function, which uniqley computes the angle between a point $(x, y)$ and the $x$ axis.

$$\theta_1(x, y) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0, y \geq 0 \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0, y < 0 \\ \frac{\pi}{2} & \text{if } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0, y < 0 \\ \text{undefined} & \text{if } x = 0, y = 0 \end{cases}$$

$$\theta_2(x, y) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0, y \geq 0 \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0, y < 0 \\ \frac{\pi}{2} & \text{if } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0, y < 0 \\ \text{undefined} & \text{if } x = 0, y = 0 \end{cases}$$

These charts sufficiently cover the circle, thus forming at atlas. We now construct two transition functions from $\mathbb{R} \to S \to \mathbb{R}$ for some point $p \in \mathbb{R}$. These overlaps occur below the $x$-axis on the left and right hand sides of the circle, between $[\pi, \pi + \alpha]$ and $[-\alpha, 2\pi]$ respectively.

$$\begin{gathered} T_1 : [\pi, \pi + \alpha] \to (x, y) \to [\pi, \pi + \alpha] \\ T_1(p) = \theta_1\left(\theta_2^{-1}(p)\right) = \theta_2\left(\theta_1^{-1}(p)\right) = p \end{gathered}$$

$$\begin{gathered} T_1 : [-\alpha, 2\pi] \to (x, y) \to [-\alpha, 2\pi] \\ T_1(p) = \theta_1\left(\theta_2^{-1}(p)\right) = \theta_2\left(\theta_1^{-1}(p)\right) = p \end{gathered}$$

## 4.2 Projection Construction

Divide the circle into four coordinate patches, each of which is a half-plane:

$$\begin{aligned} U_{top} &= \{(x, y) \in S | y \geq 0\} \\ U_{bottom} &= \{(x, y) \in S | y \leq 0\} \\ U_{right} &= \{(x, y) \in S | x \geq 0\} \\ U_{left} &= \{(x, y) \in S | x \leq 0\} \end{aligned}$$

Now define four corresponding charts $\theta : U \to \mathbb{R}$ and their inverses $X^{-1} : \mathbb{R} \to U$:

$$\begin{aligned} X_{top}(x, y) &= x & X_{top}^{-1}(x) &= (x, \sqrt{1 - x^2}) \\ X_{bottom}(x, y) &= x & X_{bottom}^{-1}(x) &= (x, \sqrt{1 - x^2}) \\ X_{left}(x, y) &= y & X_{left}^{-1}(y) &= (\sqrt{1 - y^2}, y) \\ X_{right}(x, y) &= y & X_{right}^{-1}(y) &= (\sqrt{1 - y^2}, y) \end{aligned}$$

This sufficently covers the entire circle, forming an atlas. Where the charts overlap, we construct four transition functions $T : R \to R$ for some point $p \in \mathbb{R}$:

$$\begin{aligned} T_1(p) &= X_{right}(X_{top}^{-1}(p)) = X_{right}(p, \sqrt{1 - p^2}) = \sqrt{1 - p^2} \\ T_2(p) &= X_{right}(X_{bottom}^{-1}(p)) = X_{right}(p, \sqrt{1 - p^2}) = \sqrt{1 - p^2} \\ T_3(p) &= X_{left}(X_{bottom}^{-1}(p)) = X_{left}(p, \sqrt{1 - p^2}) = \sqrt{1 - p^2} \\ T_4(p) &= X_{left}(X_{top}^{-1}(p)) = X_{left}(p, \sqrt{1 - p^2}) = \sqrt{1 - p^2} \end{aligned}$$

Note that the order of function composition does not matter here.

## 7 Approximation of MDS Matrix Factorization

sub-matrix along the diagonal is the same dissimilarity matrix via paper.

## 9 Dimensionality Reduction Bakeoff

I examine four datasets – two from ps1 and two from ps2 – and subsequently subsample them at two different rates. I then run four dimensionality reduction algorithms – MDS, IsoMap, LLE, and Laplacian Eigenmaps – on each dataset.

When plotting the first dataset 4.2 4.2, all algorithms perform quite well – even when subsampled – except for Laplacian Eigenmaps.

On the second dataset 4.2 4.2, all algorithms perform about the same. However, this dataset appears to contain random noise so there is not much to cluster. Interestingly, Laplacian Eigenmaps seems to find some slight (non-existant) structure in the data, embedding it spherically instead of uniformly.

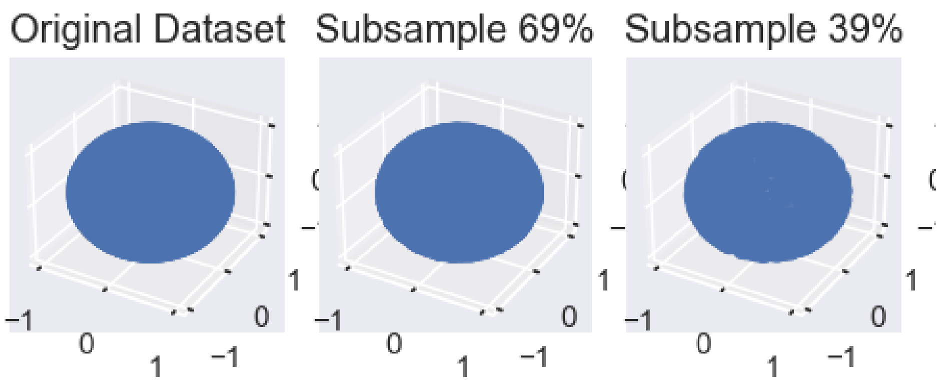
On the third dataset 4.2 4.2, all algorithms perform quite well. This dataset is a 3D sphere, which most algorithms preserve well even when subsampling the data. Once more, however, Laplacian Eigenmaps performs worse as it deforms the circle.

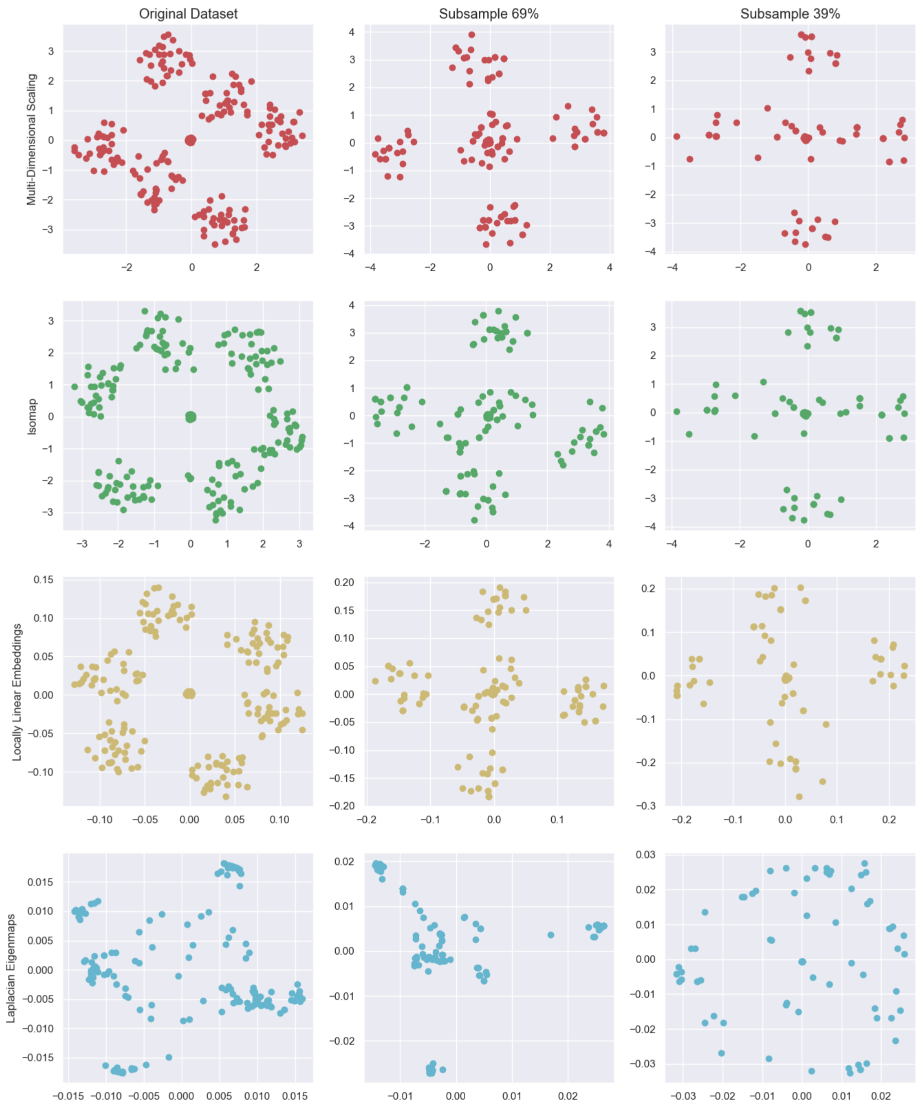
On the fourth dataset 4.2 4.2, once more all dimensionality reduction techniques perform well except for Laplacian Eigenmaps. When using the entire dataset, the dimensionality reduction algorithms embedd the data as two circles. However, when subsampling to lower and lower resolutions, the dimensionality reduction algorithms pinch one circle into a skinny band with no center (which is no longer recognizable as a circle).

Ultimately, Laplacian Eigenmaps was the worst performing dimensionality reduction technique from visual inspection. However, both Laplacian Eigenmaps and LLE suffer from large sensitivity to hyper-paramter changes. This may drastically and unpredictably alter the performance of these algorithms.

### 9.1 Dataset 1

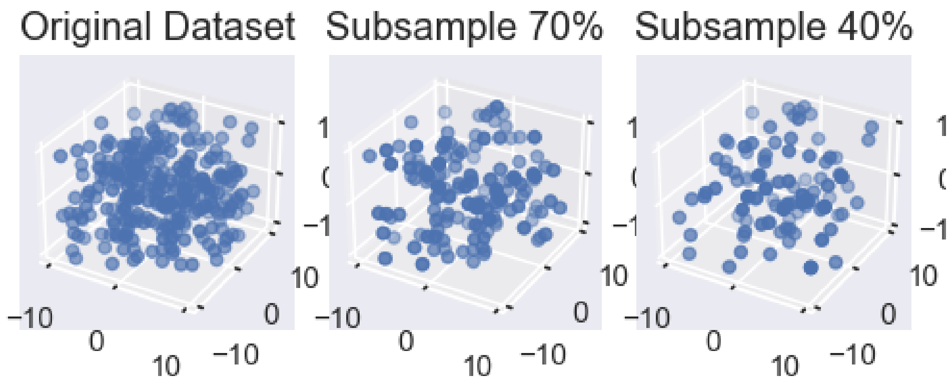
The first dataset I examine is ps1-clustering.txt. I first plot the original dataset and then plot different dimensionality reduction algorithms.

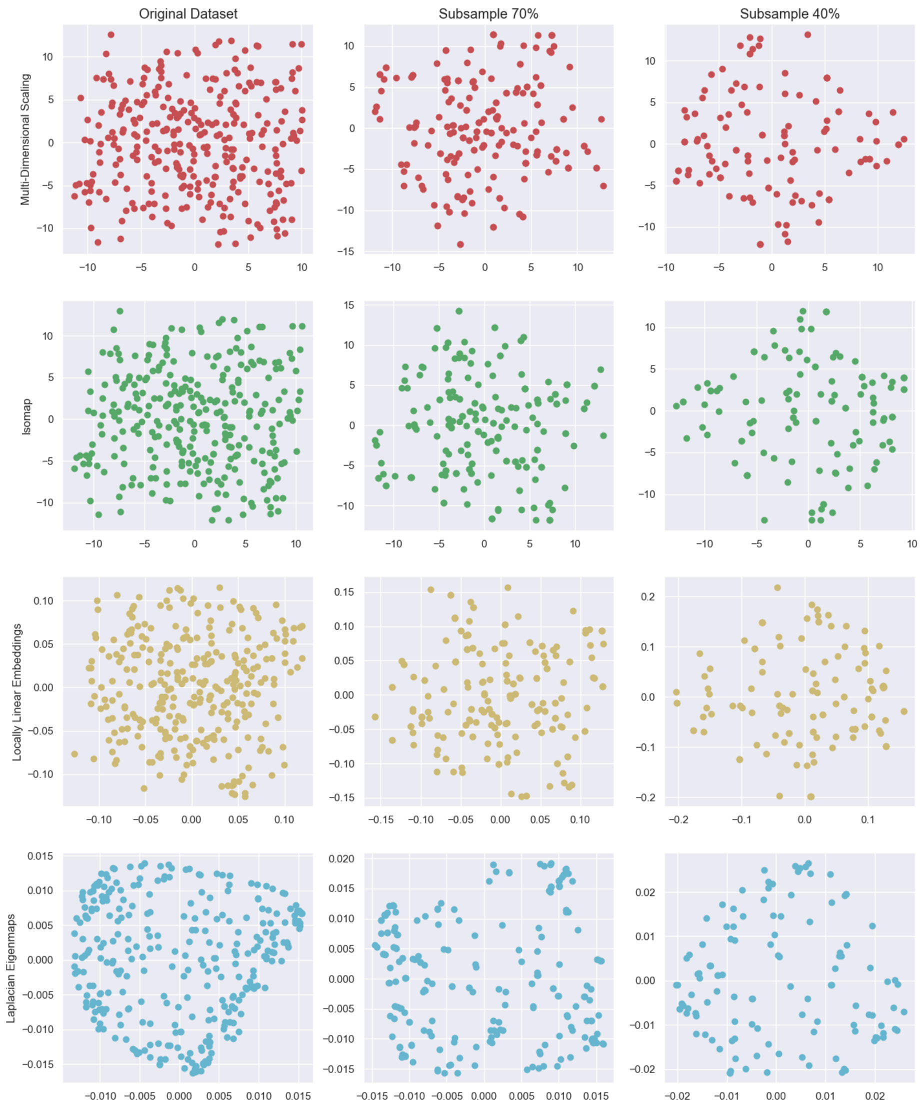

Original Dataset
Subsample 69%
Subsample 39%
−1
0
1
−1
0
1


Original Dataset
Subsample 69%
Subsample 39%
Multi-Dimensional Scaling
Isomap
Locally Linear Embeddings
Laplacian Eigenmaps

### 9.2 Dataset 2

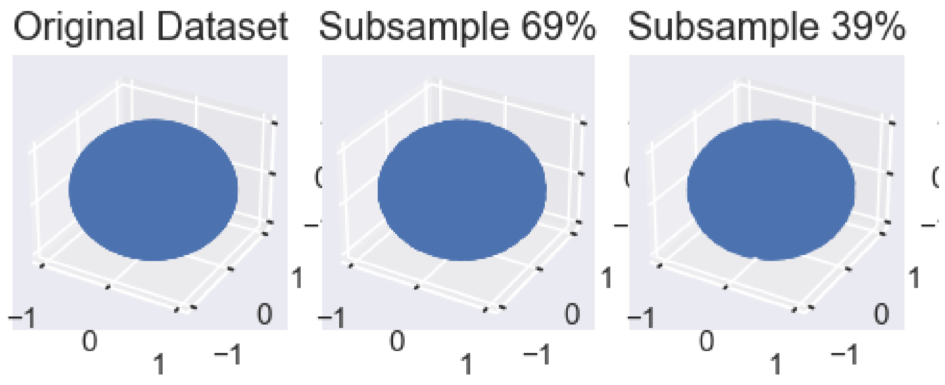
The second dataset I examine is ps1-data.txt. I first plot the original dataset and then plot different dimensionality reduction algorithms.

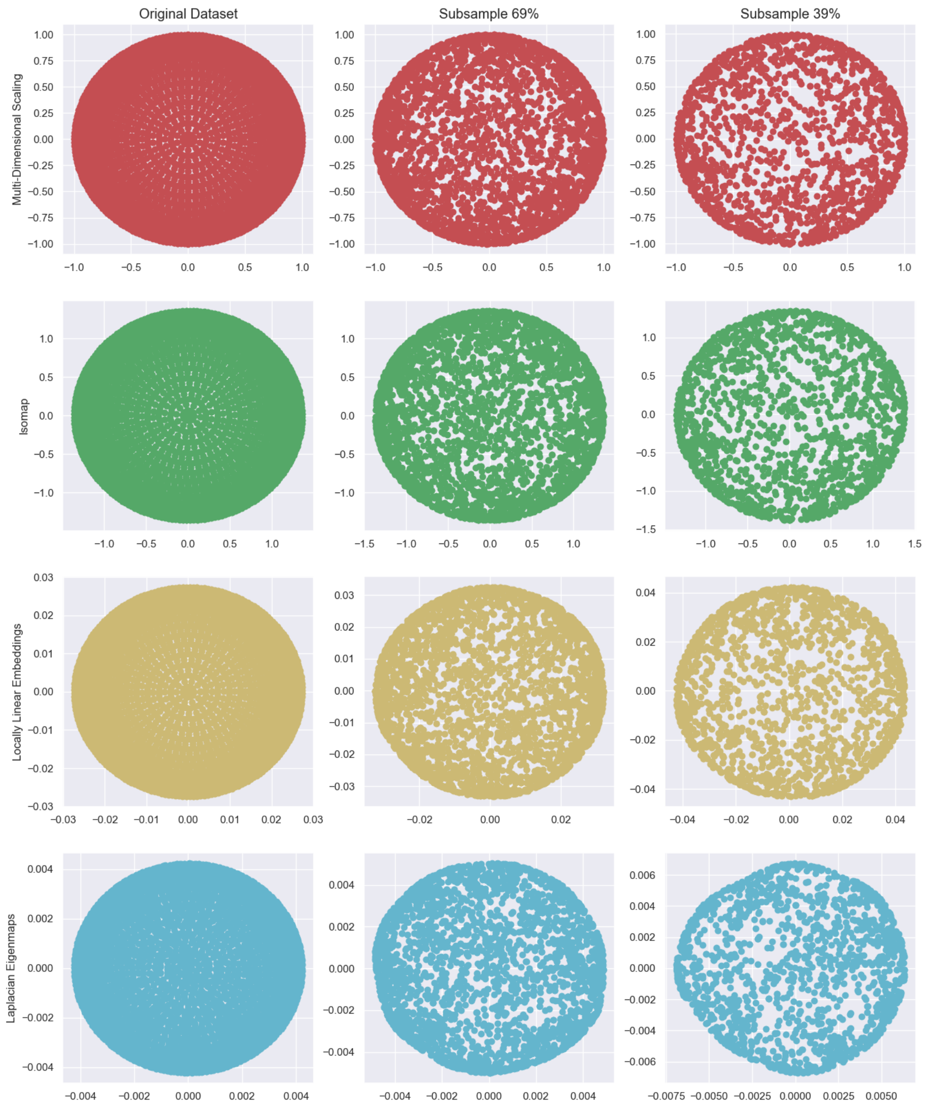



Original Dataset
Subsample 70%
Subsample 40%
Multi-Dimensional Scaling
Isomap
Locally Linear Embeddings
Laplacian Eigenmaps

## 9.3 Dataset 3

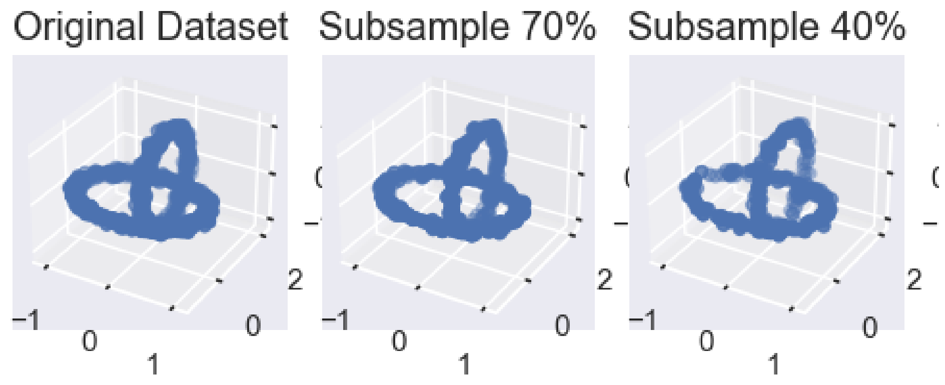
The second dataset I examine is ps2-data-1.txt. I first plot the original dataset and then plot different dimensionality reduction algorithms.

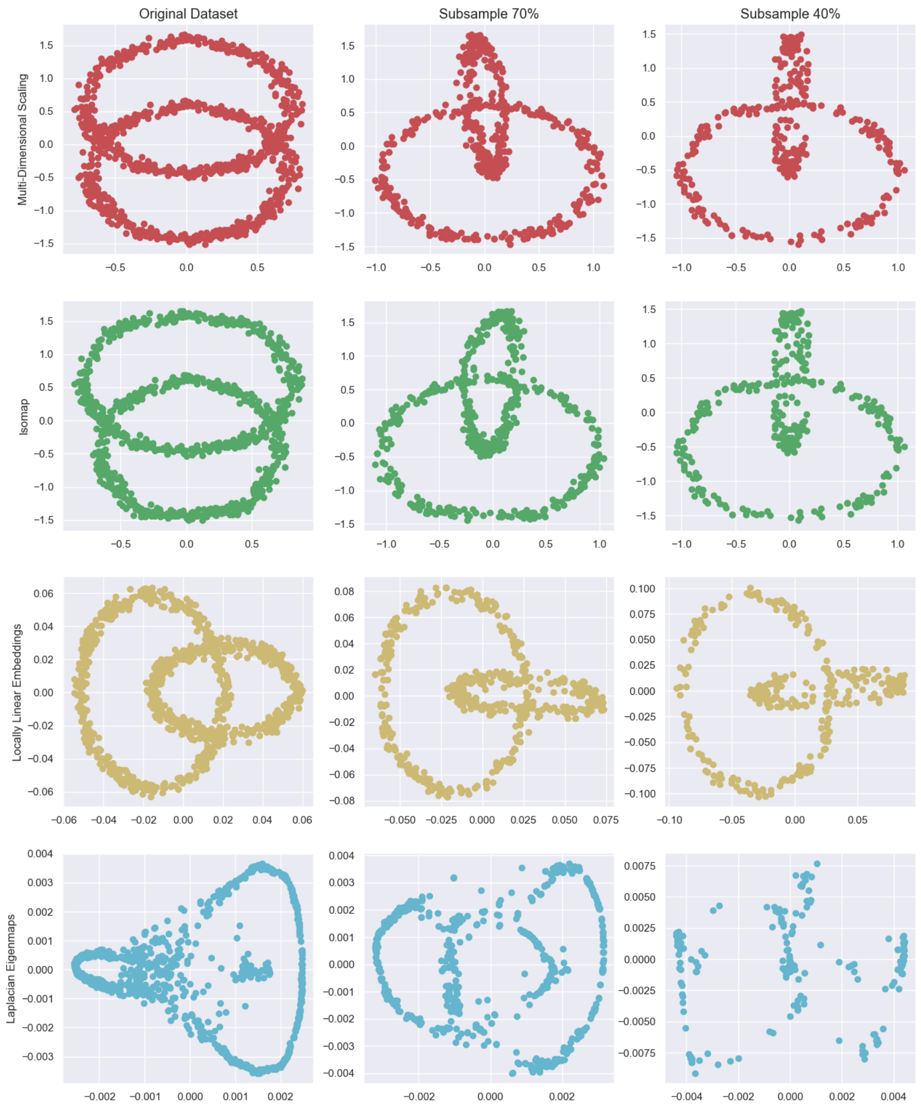



Original Dataset
Subsample 69%
Subsample 39%
Multi-Dimensional Scaling
Isomap
Locally Linear Embeddings
Laplacian Eigenmaps

## 9.4 Dataset 4

The second dataset I examine is ps2-data-2.txt. I first plot the original dataset and then plot different dimensionality reduction algorithms.




Original Dataset
Subsample 70%
Subsample 40%
Multi-Dimensional Scaling
Isomap
Locally Linear Embeddings
Laplacian Eigenmaps