Non-Linear Dimension Reduction

Inference of High Dimensional Data Project

Debepsita Mukherjee, Rohan Shinde, Sampurna Mondal, Yash Gupta

1 Introduction

Dimensionality reduction is a critical task in machine learning and data analysis, especially when dealing with high-dimensional data. The goal is to project the high-dimensional data into a lower-dimensional space while preserving as much of the relevant information and structure as possible. This course project explores various linear and non-linear dimensionality reduction techniques and evaluates their performance on the Yale Face Expression dataset.

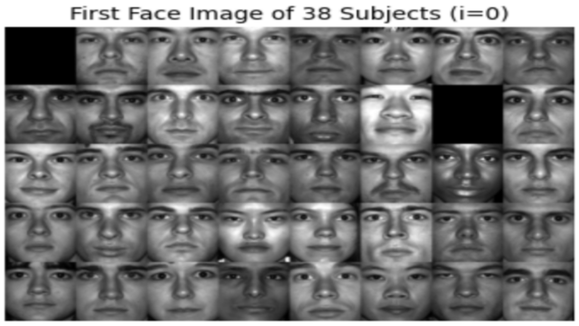


Figure 1: Subjects of Yale Image dataset

The Yale Face Expression dataset consists of $2,414$ frontal-face grayscale images of 38 subjects under different lighting conditions and facial expressions. Each image has $32,200$ pixels ($192x168$), making it a high-dimensional dataset well-suited for evaluating dimensionality reduction methods. We begin by examining the classic linear technique, Principal Component Analysis (PCA). While PCA performs reasonably well on this dataset, capturing around 90% of the variance in the top 100 principal components, there are limitations to using a linear projection for such complex, high-dimensional data. To address the non-linear nature of the data, we explore several non-linear dimensionality reduction approaches. These include Kernel PCA, which extends PCA using the kernel trick to compute principal components in a higher-dimensional transformed space. We also investigate Multidimensional Scaling (MDS), which aims to preserve pairwise distances between samples in the lower-dimensional embedding. Building on MDS, we examine Isometric Feature Mapping (Isomap), a non-linear technique that estimates geodesic distances on a weighted neighborhood graph to better represent the true multi-dimensional manifold structure. We then discuss methods focused on preserving local neighborhoods, such as Locally Linear Embedding (LLE), Diffusion Maps, and t-Stochastic Neighborhood Embedding (t-SNE). Through visualizations on synthetic datasets and evaluations on the real-world Yale Face data, we compare and contrast the strengths and limitations of each dimensionality reduction approach. Finally, we explore the challenges of applying these methods to high-dimensional real-world data and touch on strategies like estimating the intrinsic dimension. Overall, this project provides a comprehensive analysis of dimensionality reduction for high-dimensional data, with a focus on facial image data.

2 Literature Review

Dimensionality reduction has been an active area of research for several decades, driven by the increasing prevalence of high-dimensional data in fields like computer vision, bioinformatics, and signal processing. Principal Component Analysis (PCA) [Jolliffe (1986)], a classical linear technique, has been widely used but suffers from limitations in handling non-linear manifold structures common in real-world data. To address these limitations, kernel methods that implicitly compute high-dimensional mappings have gained popularity, leading to the development of Kernel PCA [SchSlkopf et al. (1997)]. Other prominent non-linear approaches include Multidimensional Scaling (MDS) [Torgerson (1952), Kruskal (1964)], which aims to preserve pairwise distances in the low-dimensional representation. Building on MDS, techniques like Isomap [Tenenbaum et al. (2000)] estimate geodesic distances on the manifold to better capture the global geometry. Methods focused on preserving local neighborhoods, such as Locally Linear Embedding (LLE) [Roweis and Saul (2000)], Diffusion Maps [Coifman and Lafon (2005)] and t-SNE [van der Maaten and Hinton (2008)], have also been widely explored.

3 Kernel PCA

Kernel PCA extends classical PCA to non-linear dimensionality reduction by mapping data into a higher-dimensional feature space using a kernel function. This process involves:

1. **Compute the kernel matrix:** Calculate the kernel matrix $\mathbf{K}$ using data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. The kernel matrix is $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, where $k(\cdot, \cdot)$ is the kernel function.

2. **Solve the eigenvalue problem:** Find principal components by solving $\mathbf{K}\boldsymbol{\alpha} = \lambda \mathbf{N}\boldsymbol{\alpha}$, where $\mathbf{N}$ is a centering matrix.

3. **Compute projections:** The projections of new data points are calculated using $\langle \mathbf{v}_r, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{N} \alpha_{ri} k(\mathbf{x}_i, \mathbf{x})$.

Kernel PCA enables non-linear mappings with various kernels, such as the Gaussian RBF kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$. The choice of kernel function and its parameters can significantly impact the performance of Kernel PCA, as it determines the feature space and the complexity of the non-linear mapping. In practice, the Gaussian RBF kernel is a popular choice due to its ability to capture complex data structures and its infinite dimensionality. However, Kernel PCA may still struggle with highly convoluted manifolds or data with complex non-linear relationships that cannot be effectively captured by the chosen kernel function. Additionally, the computational complexity of Kernel PCA can be higher than linear techniques, as it involves computing and manipulating the kernel matrix, which can be prohibitive for large datasets.

Despite these limitations, Kernel PCA has proven to be a powerful tool for non-linear dimensionality reduction and has been widely applied in various domains, including computer vision, signal processing, and bioinformatics.

4 Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a non-linear dimensionality reduction technique that aims to preserve pairwise distances between data points in a lower-dimensional space. Unlike PCA, MDS minimizes the stress or error between original high-dimensional distances and those in the embedding. The MDS algorithm involves:

1. **Compute the distance matrix:** Calculate the distance matrix $\mathbf{D}$, where D_{ij} represents the distance between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the original space.

2. **Optimize the stress function:** Find a lower-dimensional configuration $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ that minimizes the stress function:

$$\text{Stress} = \sqrt{\frac{\sum_{i<j} (D_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2}{\sum_{i<j} D_{ij}^2}}$$

   Other stress functions, such as the Sammon mapping or the Kruskal stress, can also be used depending on the specific application and data characteristics.

3. **Iterative optimization:** Use algorithms like SMACOF or gradient descent to iteratively update $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ until the stress function converges.

MDS can handle non-metric distances, making it suitable for various data types beyond Euclidean spaces, such as psychological studies and linguistic analyses, or data mining applications. However, its computational complexity can be high for large datasets (as it involves computing and optimizing over the full distance or dissimilarity matrix, which can become computationally expensive for high-dimensional data or large sample sizes), and preserving global structure can be challenging for highly non-linear manifolds. Despite these limitations, its ability to handle non-metric distances and its interpretability make it a valuable tool MDS is widely used for data visualization, exploratory data analysis, and as a foundation for techniques like Isomap, which estimates geodesic distances on the manifold to better capture its global geometry.

5 Isometric Feature Mapping (Isomap)

While MDS aims to preserve pairwise distances or dissimilarities, it still operates in the original ambient space where Euclidean distances may not accurately reflect the true geodesic distances along the low-dimensional manifold.

The notion of geodesic distance addresses this issue by estimating the shortest path or curve between two points that lies entirely on the manifold surface. Unlike straight-line Euclidean distances that can cut across the manifold, geodesic distances follow the natural geometry and topology of the data. Formally, given a manifold $\mathcal{M}$ embedded in a higher-dimensional space, the geodesic distance $d_{\mathcal{M}}(x, y)$ between two points x and y on the manifold is defined as the length of the shortest curve connecting x and y that lies entirely within $\mathcal{M}$. Preserving these geodesic distances in the low-dimensional embedding is crucial for accurately capturing the global non-linear structure of the data. However, directly computing geodesic distances is often intractable, especially when the manifold geometry is unknown. Isomap addresses this by approximating geodesic distances through a neighborhood graph:

1. **Construct a neighborhood graph:** Represent data points as nodes in a weighted graph. Edges are formed between neighboring points with weights as the Euclidean distances.

2. **Estimate geodesic distances:** Compute shortest paths between points on the graph using algorithms like Dijkstra's or Floyd's to estimate geodesic distances as below.

Algorithm 1 Floyd's Algorithm for Geodesic Distance Estimation in Isomap

```
1: Input: Distance matrix $D$ where $D[i][j]$ represents the pairwise distances. If no direct edge
   between $i$ and $j$, set $D[i][j] = \infty$.
2: Output: Matrix $D$ with geodesic distances between all pairs of points.
3: for each $k$ from 1 to $n$ do
4:     for each $i$ from 1 to $n$ do
5:         for each $j$ from 1 to $n$ do
6:             $D[i][j] = \min(D[i][j], D[i][k] + D[k][j])$
7:         end for
8:     end for
9: end for
```

3. **Apply classical MDS:** Use the geodesic distance matrix to find a low-dimensional embedding that preserves these distances.

Isomap can effectively capture the global geometry of non-linear manifolds, unlike PCA and classical MDS, which can only operate based on linear projections or pairwise distances in the ambient space.. It has been successfully used on various datasets, such as the "Swiss Roll" as shown in section 10. However, its performance can be sensitive to parameters like the neighborhood size and can degrade for manifolds with dense foldovers or intersections.

6 Local Linear Embedding (LLE)

While techniques like Isomap focus on preserving global geometric properties of the manifold through geodesic distances, another class of non-linear dimensionality reduction methods aim to retain local neighborhood information in the low-dimensional embedding. LLE is a non-linear dimensionality reduction method that preserves local neighborhood information in the low-dimensional space. The key insight behind LLE is that even if the high-dimensional data lies on a complex non-linear manifold, it can be well-approximated by piecing together linear patches within small local neighborhoods. In other words, the manifold can be viewed as being locally linear, even if it exhibits non-linear structure globally. LLE leverages this local linearity assumption by computing linear reconstructions that optimally represent each data point from its nearest neighbors. These linear coefficient weights capturing the reconstruction relationships are then transferred to the low-dimensional embedding, with the goal of preserving the same neighborhood-based reconstruction quality. The LLE algorithm consists of three main steps:

1. **Local Neighborhood Selection:** Identify k nearest neighbors for each data point $\mathbf{X}_i$ based on a distance metric.

2. **Linear Weight Computation:** Compute weights W_{ij} that linearly reconstruct $\mathbf{X}_i$ from its neighbors by minimizing the reconstruction error $\|\mathbf{X}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{X}_j\|^2$, with weights summing to 1.

3. **Embedding Computation:** Find low-dimensional representations $\mathbf{Y}_i$ that preserve these weights by minimizing $\Phi(\mathbf{Y}) = \sum_i \left\| \mathbf{Y}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{Y}_j \right\|^2$.

The low-dimensional vectors $\mathbf{Y}_i$ are obtained as the bottom $d+1$ eigenvectors of $(I-\mathbf{W})^T(I-\mathbf{W})$, excluding the smallest eigenvector. LLE efficiently maps new points out-of-sample using the learned weights by simply applying the linear reconstruction weights to the existing embedding vectors. It excels in tasks like face manifold learning and speech analysis, however, being purely local, it can potentially struggle to preserve large global structures.Numerous extensions to LLE have been proposed, such as Hessian LLE which incorporates second-order information, Manifold Charting which learns eigenfunctions instead of eigenvectors, and variants that modify the weight computation procedure.

7 Diffusion Maps

Diffusion Maps captures the intrinsic geometry and local structure of high-dimensional data by defining distances based on diffusion processes on the data manifold. Unlike Isomap, which explicitly estimate geodesic distances, Diffusion Maps implicitly capture the manifold structure by exploiting the relationship between the geometric properties of the data and the behavior of diffusion processes on the manifold. The key idea behind Diffusion Maps is to construct a Markov random walk on the data points, where the transition probabilities between points are determined by their pairwise similarities or distances. This random walk process can be interpreted as a diffusion process on the manifold, where a heat kernel defines the transition probabilities between points based on their intrinsic geometric relationships. The Diffusion Maps algorithm involves the following steps:

1. **Construct the similarity matrix:** Compute the similarity matrix $\mathbf{W}$ using a kernel function like the Gaussian kernel: $W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\epsilon}\right)$, where ϵ controls the neighborhood size and W_{ij} represent the pairwise similarities or affinities between data points $\mathbf{x}_i$ and $\mathbf{x}_j$

2. **Construct the Markov transition matrix:** Normalize $\mathbf{W}$ to obtain the Markov transition matrix $\mathbf{P}$: $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, with $\mathbf{D}$ being a diagonal matrix where $D_{ii} = \sum_j W_{ij}$.

3. **Compute the diffusion map embedding:** Find the top d non-trivial eigenvectors $\{\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_d\}$ and eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ of $\mathbf{P}$ sorted in descending order. The embedding for a data point $\mathbf{x}_i$ is $\Psi_t(\mathbf{x}_i) = \left(\lambda_1^t\psi_1(\mathbf{x}_i), \lambda_2^t\psi_2(\mathbf{x}_i), \ldots, \lambda_d^t\psi_d(\mathbf{x}_i)\right)$, where t is the diffusion time parameter.

Diffusion Maps effectively capture non-linear structures and reveal patterns in complex data by integrating local similarities over the manifold using the eigenvalues and eigenvectors of the Markov transition matrix which encodes information about the geometry and topology of the manifold, with the top eigenvectors corresponding to the most prominent geometric features. However, its performance depends on the choice of parameters like ϵ and t and inappropriate parameter choices can lead to either over-smoothing or under-smoothing of the data, resulting in poor embeddings. Despite this, Diffusion Maps is widely used in image analysis, speech recognition, and machine learning for its ability to recover the intrinsic geometry of data.

8 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a popular non-linear dimensionality reduction technique that preserves both local and global structures in high-dimensional data.

Unlike PCA or MDS, t-SNE models pairwise similarities between data points using probability distributions, making it effective for visualizing complex datasets. The key idea behind t-SNE is to model the pairwise similarities between data points in the high-dimensional space using conditional probabilities, and then minimize the divergence between these similarities and the corresponding similarities in the lower-dimensional embedding space. Specifically, t-SNE defines two probability distributions: the high-dimensional input probability distribution P and the low-dimensional embedding probability distribution Q. The t-SNE algorithm involves the following steps:

1. **High-dimensional probability distribution:** Compute the pairwise similarities between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ using a Gaussian kernel: $p_{j|i} = \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$ (σ_i is typically chosen based on the local density of the data around $\mathbf{x}_i$), with p_{ij} symmetrized as: $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$.
2. **Low-dimensional probability distribution:** Define the pairwise similarities q_{ij} in the embedding space using a heavy-tailed Student's t-distribution: $q_{ij} = \frac{(1+\|\mathbf{y}_i-\mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|\mathbf{y}_k-\mathbf{y}_l\|^2)^{-1}}$.
3. **Minimize KL divergence:** Find the low-dimensional embedding $\{\mathbf{y}_i\}$ that minimizes the Kullback-Leibler (KL) divergence between P and Q: $\mathrm{KL}(P\|Q) = \sum_{i\neq j} p_{ij}\log\frac{p_{ij}}{q_{ij}}$, typically using gradient descent.

t-SNE effectively captures both local and global data structures due to its use of conditional probabilities and the Student's t-distribution. However, it can be computationally expensive and may encounter convergence issues for large datasets. Despite these challenges, t-SNE is widely used in machine learning, bioinformatics, and image analysis for non-linear dimensionality reduction and data visualization.

9 Fractal Dimension

The fractal dimension analyzes how data points fill high-dimensional space. It is akin to how books might be scattered across many shelves (high spread-outness) or crammed onto a few (low spread-outness). This dimension measures the complexity of data distribution beyond the raw number of features by examining how data density changes with scale. Fractal dimension refers to various dimensions characterizing fractals, such as capacity, correlation, and information dimensions, unified by the q-dimension. For a random variable $\mathbf{y}$ with distribution function $F(.)$ and density function $f(.)$:

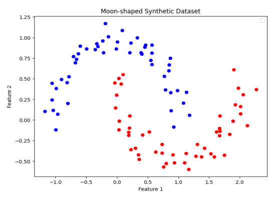
Given $\epsilon > 0$, cover the support of F with a grid of cubes of edge length ϵ. Let $N(\epsilon)$ be the number of intersecting cubes, and p_i the probability of cube i being populated. The q-dimension D_q is defined as: $D_q = \lim\limits_{\epsilon\to 0}\dfrac{\log(\sum_{i=1}^{N(\epsilon)} p_i^q)}{(q-1)\log(\epsilon)}$ if the limit exists. Setting $q = 0$ in the q-dimension formula gives the capacity dimension: $d_{cap} = D_0 = \lim\limits_{\epsilon\to 0}\dfrac{\log(N(\epsilon))}{\log(\epsilon)}$. This dimension, also known as the box-counting dimension, does not depend on p_i. **Algorithm:**

1. Overlay a grid on the high-dimensional space where data points reside, creating boxes of length ϵ. Count the minimum number of boxes $N(\epsilon)$ needed to cover all data points. Repeat for several box sizes, decreasing ϵ.

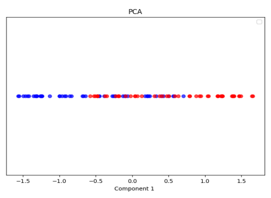
2. Plot the logarithm of both ϵ and $N(\epsilon)$ on a log-log plot (x-axis: $\log(\epsilon)$, y-axis: $\log(N(\epsilon))$).
3. The box-counting dimension is the negative slope of the log-log plot.

10 Applications to simulated data

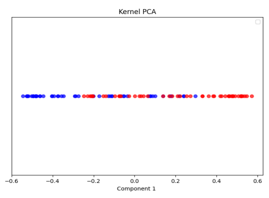
Example (a): Half moons data: Figure 2 compares all the methods on the half moons data



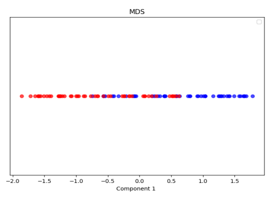
(a) Actual data



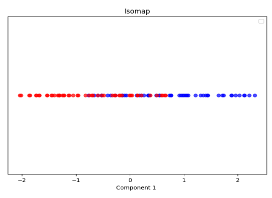
(b) PCA



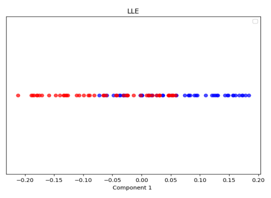
(c) KPCA



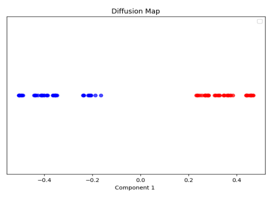
(d) MDS



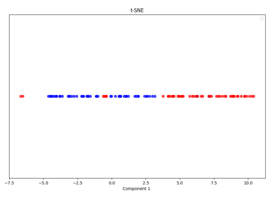
(e) ISOMAP



(f) LLE



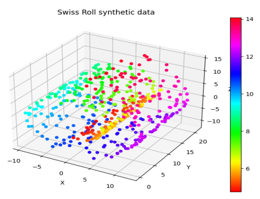
(g) Diffusion Map



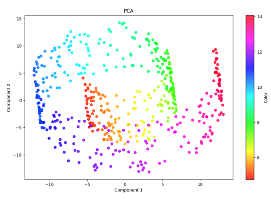
(h) t-SNE

Figure 2: Comparison among six methods for half moon data ($N = 100$)

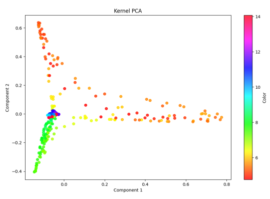
Example (b): 3D - Swiss roll:: Figure 3 compares all the methods on the Swiss Roll data



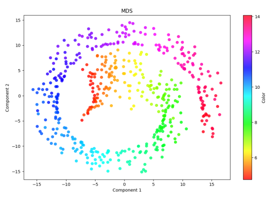
(a) Actual data



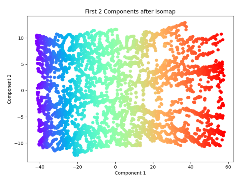
(b) PCA



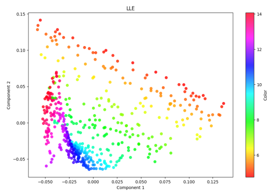
(c) KPCA



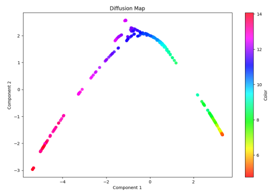
(d) MDS



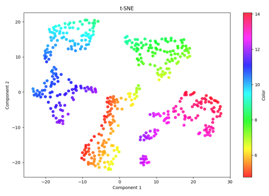
(e) ISOMAP



(f) LLE



(g) Diffusion Map



(h) t-SNE

Figure 3: Comparison among six methods for swiss roll data ($N = 500$)

11 Application to Yale face Dataset

For the Yale Face dataset the intrinsic dimension was found out to be 12. And the results are summarized in Table (1).

Methods	**Accuracy(%)**
PCA	24.637
Kernel PCA	24.637
MDS	14.285
Isomap	46.376
LLE	42.443
Diffusion maps	27.950
t-SNE	57.142

Table 1: Accuracy of 5-NN classifier for the dimension reduced data

As is evident from Table (1), the non-linear dimension reduction techniques result in accuracy that is significantly greater than that of a linear dimension reduction technique namely the PCA. This suggests that techniques like Isomap, LLE, and t-SNE are able to detect and extract the inherent important non-linear structures of the data better than PCA.

References

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 2005.

I. T. Jolliffe. *Principal Component Analysis*. Springer New York, NY, 1986.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 1964.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.

B. SchSlkopf, A. Smola, and K.-R. Mfiller. Kernel principal component analysis. 1997.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 1952.

L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

Contributions of each member: Every member was involved in the simulations and data analysis work. Other major contributions:

1. **Debepsita Mukherjee**: Fractal Dimensions and Applications to Yale Face Data

2. **Rohan Shinde**: Kernel PCA and MDS

3. **Sampurna Mondal**: Isomap and LLE

4. **Yash Gupta**: Diffusion Maps and t-SNE