High Dimensional Inference Course Project

Debepsita Mukherjee
Rohan Shinde
Sampurna Mondal
Yash Gupta

Indian Statistical Institute

20th April, 2024

Yale Face Expression Data

- 2414 frontal-face images of over 38 subjects

Yale Face Expression Data

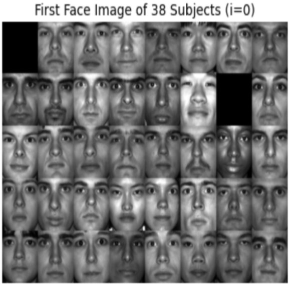
- 2414 frontal-face images of over 38 subjects
- 64 images per subject under different lighting conditions and various facial expressions

Yale Face Expression Data

- 2414 frontal-face images of over 38 subjects
- 64 images per subject under different lighting conditions and various facial expressions
- $\#$ of Features/pixels $= 192 \times 168 \approx 32200$

Yale Face Expression Data

- 2414 frontal-face images of over 38 subjects
- 64 images per subject under different lighting conditions and various facial expressions
- $\#$ of Features/pixels $= 192 \times 168 \approx 32200$



How does PCA fare here?

- PCA: Linear dimensionality reduction by projecting data onto directions of maximum variance

How does PCA fare here?

- PCA: Linear dimensionality reduction by projecting data onto directions of maximum variance

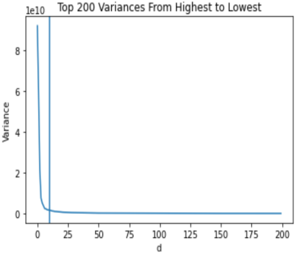


Figure 1: Scree Plot for PCA

How does PCA fare here?

- PCA: Linear dimensionality reduction by projecting data onto directions of maximum variance

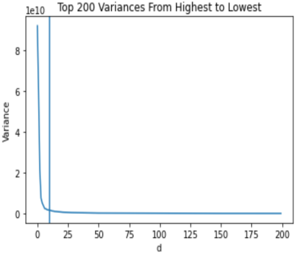


Figure 1: Scree Plot for PCA

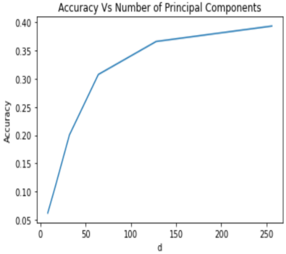


Figure 2: Accuracy of PCA for different intrinsic dimensions using KNN classifier

How does PCA fare here?

- PCA: Linear dimensionality reduction by projecting data onto directions of maximum variance

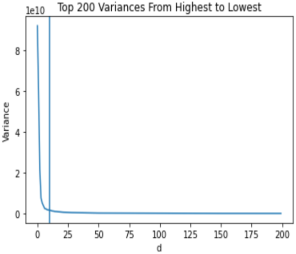


Figure 1: Scree Plot for PCA

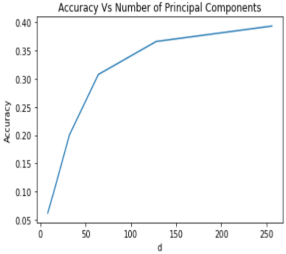


Figure 2: Accuracy of PCA for different intrinsic dimensions using KNN classifier

- Images will be locally similar in many regions and symmetric in few dimensions

How does PCA fare here?

- PCA: Linear dimensionality reduction by projecting data onto directions of maximum variance

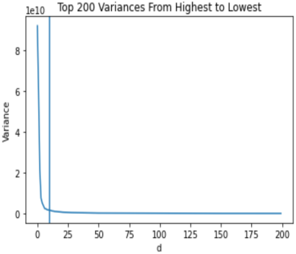


Figure 1: Scree Plot for PCA

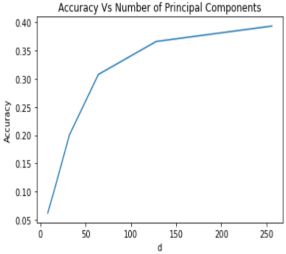
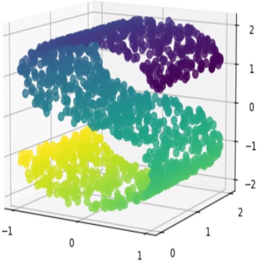


Figure 2: Accuracy of PCA for different intrinsic dimensions using KNN classifier

- Images will be locally similar in many regions and symmetric in few dimensions
- Raises concern over the *linear* projection of PCA

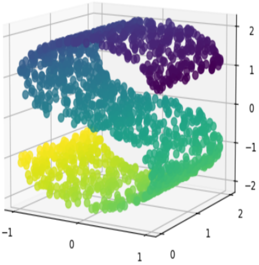
Non-Linear Dimension Reduction

Why Non-Linear Dimension Reduction



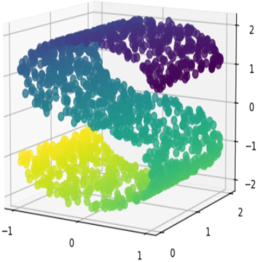
- Direction of maximum variation may not be *linear* especially in high-dimensional data

Why Non-Linear Dimension Reduction



- Direction of maximum variation may not be *linear* especially in high-dimensional data
- **Preserving global/local topology** may be of importance

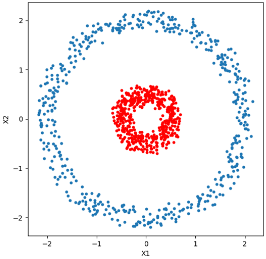
Why Non-Linear Dimension Reduction



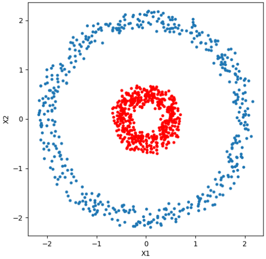
- Direction of maximum variation may not be *linear* especially in high-dimensional data
- **Preserving global/local topology** may be of importance
- Captures **complex non-linear relationships** among the variables

Extending Concept of PCA

Kernel PCA

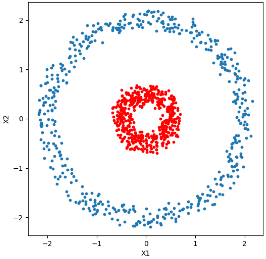


Kernel PCA



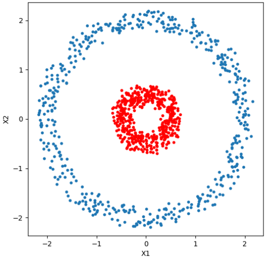
- Apply high dimensional transformation to capture non-linearity

Kernel PCA



- Apply high dimensional transformation to capture non-linearity
- Use Kernel trick to compute covariance matrix in transformed space (Gram matrix)

Kernel PCA



- Apply high dimensional transformation to capture non-linearity
- Use Kernel trick to compute covariance matrix in transformed space (Gram matrix)
- Apply PCA on the covariance matrix of the transformed data

Kernel PCA

Kernel PCA Algorithm (SchSlkopf et al., 1997)

1. Consider $\Phi : \mathbb{R}^p \to \mathcal{H}$ ($\dim(\mathcal{H}) > p$), assume $\sum_{i=1}^N \Phi(\mathbf{x}_i) = 0$

Kernel PCA

Kernel PCA Algorithm (SchSlkopf et al., 1997)

1. Consider $\Phi : \mathbb{R}^p \to \mathcal{H}$ $(\dim(\mathcal{H}) > p)$, assume $\sum_{i=1}^{N} \Phi(\mathbf{x}_i) = 0$
2. For known kernel function $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$, define $K = ((K_{ij}))$, with $K_{ij} = \Phi(\mathbf{X}_i)^T \Phi(\mathbf{X}_j)$ $(\mathbf{X}_1, \cdots, \mathbf{X}_N$ are the observations)

Kernel PCA

Kernel PCA Algorithm (SchSlkopf et al., 1997)

1. Consider $\Phi : \mathbb{R}^p \to \mathcal{H}$ ($\dim(\mathcal{H}) > p$), assume $\sum_{i=1}^{N} \Phi(\mathbf{x}_i) = 0$
2. For known kernel function $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$, define $K = ((K_{ij}))$, with $K_{ij} = \Phi(\mathbf{X}_i)^T \Phi(\mathbf{X}_j)$ ($\mathbf{X}_1, \cdots, \mathbf{X}_N$ are the observations)
3. Directly compute the projections from a point in the feature space $\Phi(\mathbf{x})$ onto the r-th principal component (V^r) as:
   $(V^r)^T \Phi(\mathbf{x}) = \left(\sum_{i=1}^{N} a_i^r \Phi(\mathbf{x_i})\right)^T \Phi(\mathbf{x})$ where a_i^r are obtained by solving
   1. Eigenvector equation: $N\lambda \mathbf{a} = K\mathbf{a}$
   2. Normalizing eigenvector equation: $(V^r)^T V^r = 1$

PCA vs Kernel PCA

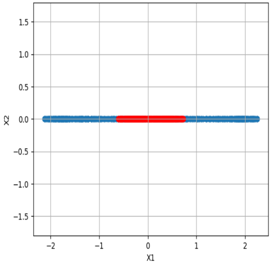


Figure 3: Dimension reduction via PCA

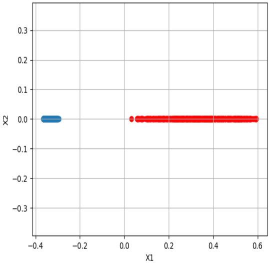


Figure 4: Dimension reduction via PCA

- For *choice of kernel* refer Bernhard et al. (1998)

Multidimensional Scaling (MDS)

- Measure all pairwise Euclidean distances between samples

Multidimensional Scaling (MDS)

- Measure all pairwise Euclidean distances between samples
- Embed in lower dimensional space preserving pairwise distances

Multidimensional Scaling (MDS)

- Measure all pairwise Euclidean distances between samples
- Embed in lower dimensional space preserving pairwise distances

MDS Algorithm

1. Given a symmetric distance or affinity matrix D, set up the squared proximity matrix $\mathbf{D}^{(2)} = [d_{ij}^2]$.
2. Define $\mathbf{B} = -\frac{1}{2}\mathbf{H}\mathbf{D}^{(2)}\mathbf{H}$ where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}_N\mathbf{1}_N^T$
3. $\mathbf{Y} = \mathbf{E}_m\mathbf{\Lambda}_m^{1/2}$ where $\mathbf{\Lambda}_m$ is the diagonal matrix of m largest eigenvalues of $\mathbf{B}$ and $\mathbf{E}_m$ is the matrix of the respective m eigenvectors.

Multidimensional Scaling (MDS)

- Measure all pairwise Euclidean distances between samples
- Embed in lower dimensional space preserving pairwise distances

MDS Algorithm

1. Given a symmetric distance or affinity matrix D, set up the squared proximity matrix $\mathbf{D}^{(2)} = [d_{ij}^2]$.
2. Define $\mathbf{B} = -\frac{1}{2}\mathbf{H}\mathbf{D}^{(2)}\mathbf{H}$ where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}_N\mathbf{1}_N^T$
3. $\mathbf{Y} = \mathbf{E}_m\mathbf{\Lambda}_m^{1/2}$ where $\mathbf{\Lambda}_m$ is the diagonal matrix of m largest eigenvalues of $\mathbf{B}$ and $\mathbf{E}_m$ is the matrix of the respective m eigenvectors.

$$\Updownarrow$$

$$\arg\min_{\mathbf{Y}_1,\cdots,\mathbf{Y}_N} \sum_{i=1}^{N}\sum_{j=1}^{N}(D_{ij} - ||\mathbf{Y}_i - \mathbf{Y}_j||)^2$$

Connection b/w Kernel PCA & MDS

Result (Williams, 2002)

Using an Isotropic Kernel function the Kernel PCA can be interpreted as performing a kind of MDS.

Isotropic Kernel: Kernel depending only on the Euclidean distance

Other notions of Distances

Geodesic Distance

Geodesic Distance

Assumption:
Euclidean distance is “meaningful” for short distances

Geodesic Distance

Assumption:
Euclidean distance is “meaningful” for short distances

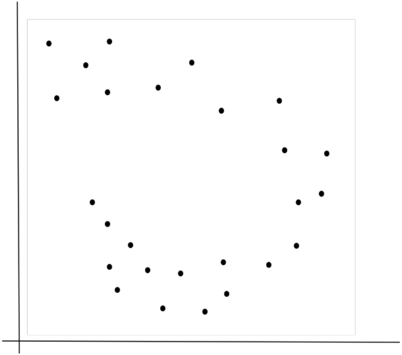


Figure 5: Geodesic Distance

Geodesic Distance

Assumption:
Euclidean distance is "meaningful" for short distances

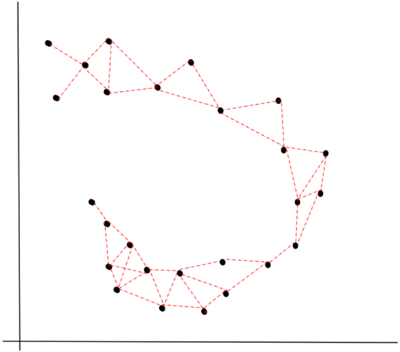


Figure 5: Geodesic Distance

Geodesic Distance

Assumption:
Euclidean distance is “meaningful” for short distances

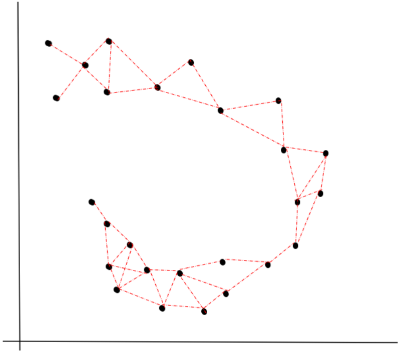


Figure 5: Geodesic Distance

Geodesic Distance: Shortest distance in this graph

Assumption:
Euclidean distance is "meaningful" for short distances

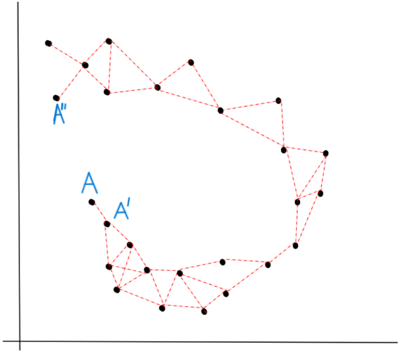


Figure 5: Geodesic Distance

Geodesic Distance: Shortest distance in this graph

Isometric Feature Mapping (ISOMAP)

1. We have a training set

Isometric Feature Mapping (ISOMAP)

1. We have a training set
2. Calculate *geodesic distance* for each pair

Isometric Feature Mapping (ISOMAP)

1. We have a training set
2. Calculate *geodesic distance* for each pair
3. Use MDS to embed corresponding points to a new space

Isometric Feature Mapping (ISOMAP)

1. We have a training set
2. Calculate *geodesic distance* for each pair
3. Use MDS to embed corresponding points to a new space

Application on Swiss Roll

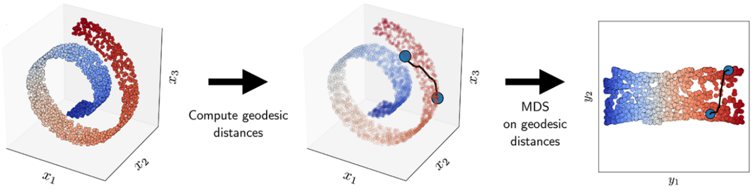


Figure 6: Applying ISOMAP to Swiss Roll data

Isometric Feature Mapping (ISOMAP)

ISOMAP Algorithm

1. Determine k neighbourhood graph G of the observed data $\{x_i\}$
2. Compute shortest paths in the graph *for all pairs* of data points to form a distance matrix D. Each edge x_i, x_j is weighted by its Euclidean length $||x_i - x_j||$ or by some other useful metric
3. Apply MDS to the resulting shortest-path distance matrix D

Compare PCA with ISOMAP

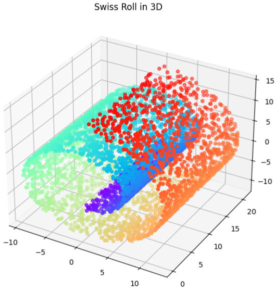


Figure 7: Swiss Roll Data with $N = 1000$

Compare PCA with ISOMAP

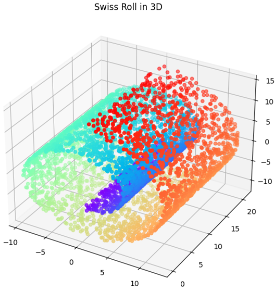


Figure 7: Swiss Roll Data with $N = 1000$

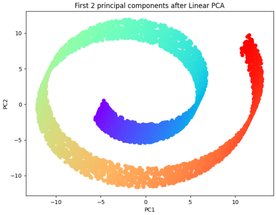


Figure 8: PCA

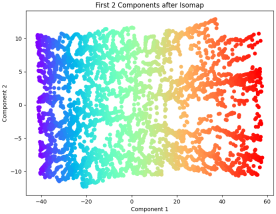


Figure 9: ISOMAP

Compare PCA with ISOMAP

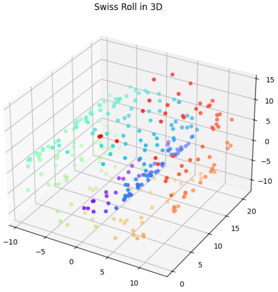


Figure 10: Swiss Roll Data with $N = 300$

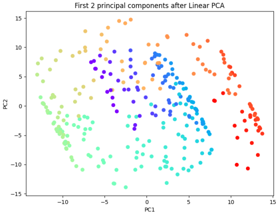


Figure 11: PCA

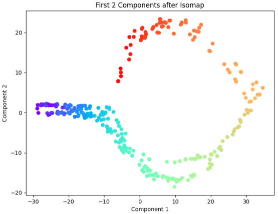


Figure 12: ISOMAP

Isometric Feature Mapping (ISOMAP)

Theorem (de Silva and Tenenbaum, 2002)

Let Y be sampled from a bounded convex region in $\mathbb{R}^p$, with respect to a density function $\alpha = \alpha(y)$. Let f be a C^2-smooth isometric embedding of that region in $\mathbb{R}^d$. Given $\lambda, \mu > 0$, for a suitable choice of neighborhood size parameter k, we have

$$1 - \lambda \leq \frac{\text{recovered distance}}{\text{original distance}} \leq 1 + \lambda$$

with probability at least $1 - \mu$, provided that the simple size is sufficiently large [The formula is taken to hold for all pairs of points simultaneously].

Isometric Feature Mapping (ISOMAP)

Theorem (de Silva and Tenenbaum, 2002)

Let Y be sampled from a bounded convex region in $\mathbb{R}^p$, with respect to a density function $\alpha = \alpha(y)$. Let f be a C^2-smooth isometric embedding of that region in $\mathbb{R}^d$. Given $\lambda, \mu > 0$, for a suitable choice of neighborhood size parameter k, we have

$$1 - \lambda \leq \frac{\text{recovered distance}}{\text{original distance}} \leq 1 + \lambda$$

with probability at least $1 - \mu$, provided that the simple size is sufficiently large [The formula is taken to hold for all pairs of points simultaneously].

- MDS is isometric C^2-smooth embedding

Isometric Feature Mapping (ISOMAP)

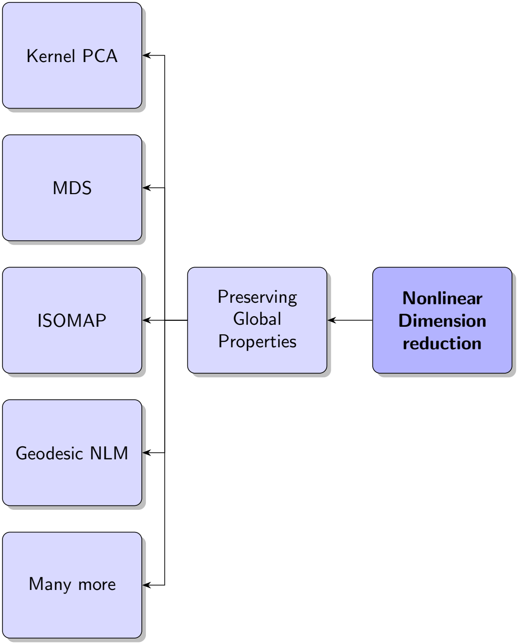
Theorem (de Silva and Tenenbaum, 2002)

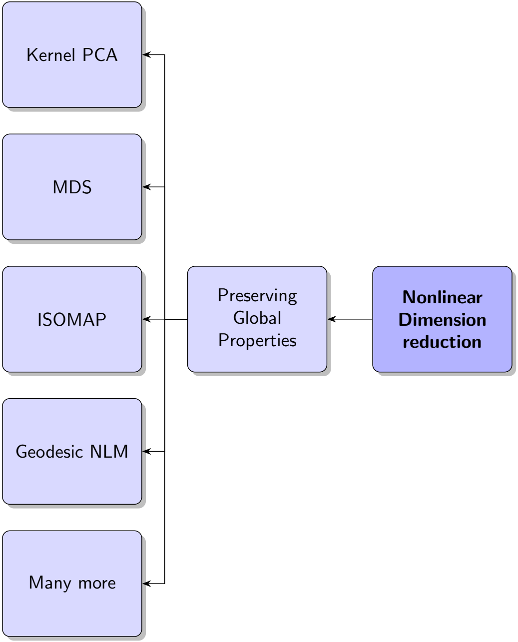
Let Y be sampled from a bounded convex region in $\mathbb{R}^p$, with respect to a density function $\alpha = \alpha(y)$. Let f be a C^2-smooth isometric embedding of that region in $\mathbb{R}^d$. Given $\lambda, \mu > 0$, for a suitable choice of neighborhood size parameter k, we have

$$1 - \lambda \leq \frac{\text{recovered distance}}{\text{original distance}} \leq 1 + \lambda$$

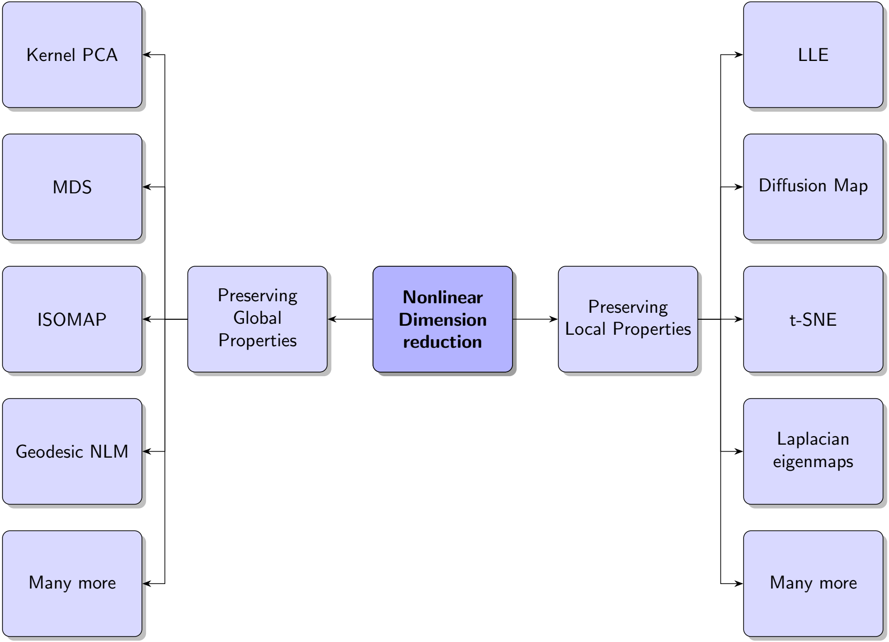
with probability at least $1 - \mu$, provided that the simple size is sufficiently large [The formula is taken to hold for all pairs of points simultaneously].

- MDS is isometric C^2-smooth embedding
- ISOMAP preserves distance (globally)


Kernel PCA
MDS
ISOMAP
Geodesic NLM
Many more
Preserving Global Properties
Nonlinear Dimension reduction



Some subregions must be locally stretched or shrunk in order to embed them in a lower-dimensional space



Some subregions must be locally stretched or shrunk in order to embed them in a lower-dimensional space

Preserving Local Properties

Locally Linear Embedding (LLE)

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 1

Build Local Models

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 1

Build Local Models

1. Use any distance metric to get the k-nearest neighbors for each point

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 1

Build Local Models

1. Use any distance metric to get the k-nearest neighbors for each point

$$N_i : \ k\text{- nearest neighbors of } i^{th} \text{ point}$$

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 1

Build Local Models

1. Use any distance metric to get the k-nearest neighbors for each point

$$N_i : \ k\text{- nearest neighbors of } i^{th} \text{ point}$$

2. For each point, identify the weighted sum of the neighbors that predicts the location of the point

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 1

Build Local Models

1. Use any distance metric to get the k-nearest neighbors for each point

$$N_i : \; k\text{- nearest neighbors of } i^{th} \text{ point}$$

2. For each point, identify the weighted sum of the neighbors that predicts the location of the point

$$\hat{\mathbf{X}}_i = \sum_{\mathbf{X}_j \in N_i} w_{ij} \mathbf{X}_j \quad \text{s.t.} \sum_{\mathbf{X}_j \in N_i} w_{ij} = 1$$

To calculate weights we minimise: $\sum_{i=1}^{N} ||\mathbf{X}_i - \hat{\mathbf{X}}_i||^2$

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 2

Embedding

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 2

Embedding

1. The same weights that reconstruct the data points in p dimensions should reconstruct it in the manifold in d dimensions

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 2

Embedding

1. The same weights that reconstruct the data points in p dimensions should reconstruct it in the manifold in d dimensions
2. To minimise:

$$\left\| \mathbf{Y}_i - \sum_{\mathbf{Y}_j \in N_i} w_{ij} \mathbf{Y}_j \right\|^2$$

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 2

Embedding

1. The same weights that reconstruct the data points in p dimensions should reconstruct it in the manifold in d dimensions
2. To minimise:

$$\left\|\mathbf{Y}_i - \sum_{\mathbf{Y}_j \in N_i} w_{ij}\mathbf{Y}_j\right\|^2$$

Equivalent to minimise:

$$(\mathbf{Y}^T(I - \mathbf{W})^T(I - \mathbf{W})\mathbf{Y}) \quad \text{subject to} \quad \frac{1}{N}\mathbf{Y}^T\mathbf{Y} = I$$

Locally Linear Embedding (LLE)

LLE Algorithm (Ghojogh et al., 2020)

Phase 2

Embedding

1. The same weights that reconstruct the data points in p dimensions should reconstruct it in the manifold in d dimensions
2. To minimise:

$$\left\| \mathbf{Y}_i - \sum_{\mathbf{Y}_j \in N_i} w_{ij} \mathbf{Y}_j \right\|^2$$

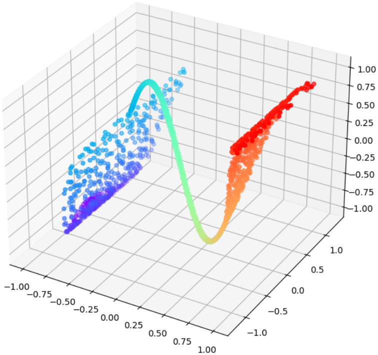
   Equivalent to minimise:

$$(\mathbf{Y}^T (I - \mathbf{W})^T (I - \mathbf{W}) \mathbf{Y}) \quad \text{subject to} \quad \frac{1}{N} \mathbf{Y}^T \mathbf{Y} = I$$

3. Solving *Lagrangian* we can get,

$$\mathbf{Y} \text{ is the } \textit{eigenvectors} \text{ of } (I - \mathbf{W})^T (I - \mathbf{W})$$

Locally Linear Embedding (LLE)



Locally Linear Embedding (LLE)

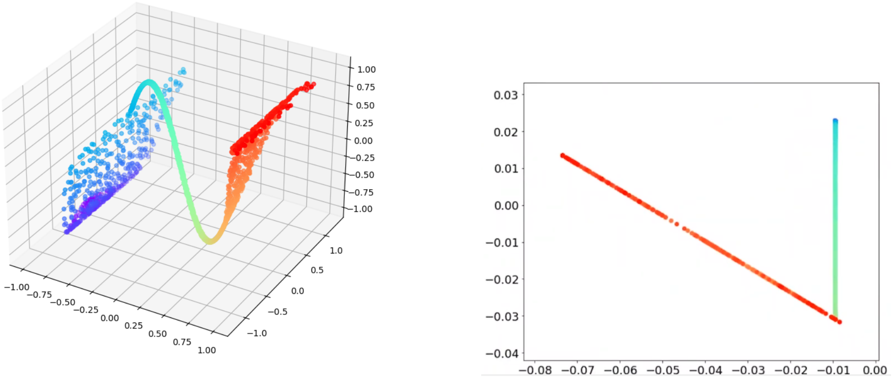


Figure 13: Applying LLE to a synthetic data

Diffusion Map

Idea: To define a notion of distance between two points such a way that we move along the structure of the data.

Diffusion Map

Idea: To define a notion of distance between two points such a way that we move along the structure of the data.

Diffusion Map Algorithm

1. Construct a transition matrix P : $P_{i,j} \propto k(x_i, x_j)$

Diffusion Map

Idea: To define a notion of distance between two points such a way that we move along the structure of the data.

Diffusion Map Algorithm

1. Construct a transition matrix P : $P_{i,j} \propto k(x_i, x_j)$
2. Calculate P^t where t is time step.

Diffusion Map

Idea: To define a notion of distance between two points such a way that we move along the structure of the data.

Diffusion Map Algorithm

1. Construct a transition matrix P : $P_{i,j} \propto k(x_i, x_j)$
2. Calculate P^t where t is time step.
3. If points i and j are locally close then $P^t_{i,k} \approx P^t_{j,k}$ for all points k.
4. Diffusion Distance : $D_t(x_i, x_j)^2 = \Sigma_k \mid (P^t_{i,k} - P^t_{j,k}) \mid^2$
5. Which is Euclidean distance between two points Y_i & Y_j in the diffusion space where $Y_i = (P^t_{i,1}, ..., P^t_{i,n})$.

Diffusion Map

Idea: To define a notion of distance between two points such a way that we move along the structure of the data.

Diffusion Map Algorithm

6. Compute the d largest eigenvalues of P^t and the corresponding eigenvectors.
7. We can leave smaller eigenvalues to get embedding Ψ_t where in low dimensional space $\Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \ldots, \lambda_d^t \psi_d(x))$ where λ_i & $\psi_i(x)$ are eigenvalue and eigenvector of P_t.
8. Thus we get the diffusion map from the original data to a d-dimensional space which is embedded in the original space.

t-Stochastic Neighbourhood Embedding (t-SNE)

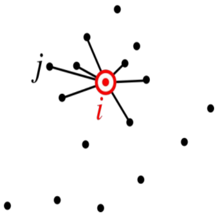
Minimize the KL divergence between high and low dimensional affinities p_{ij} & q_{ij}

t-Stochastic Neighbourhood Embedding (t-SNE)

Minimize the KL divergence between high and low dimensional affinities p_{ij} & q_{ij}

$$\mathcal{L} = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

High penalty for putting close neighbour far away



t-Stochastic Neighbourhood Embedding (t-SNE)

t-SNE Algorithm

1. High-dimensional similarities:

$$p_{j|i} = \frac{\exp\left(- \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(- \left\| \mathbf{x}_i - \mathbf{x}_k \right\|^2 / 2\sigma_i^2\right)}$$

2. Then symmetrize and normalize to sum to one: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$
3. Low-dimensional similarities:

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = k\left(\left\| \mathbf{y}_i - \mathbf{y}_j \right\|\right), \quad Z = \sum_{k \neq l} w_{kl}$$

t-Stochastic Neighbourhood Embedding (t-SNE)

t-SNE Algorithm

1. High-dimensional similarities:

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma_i^2\right)}$$

2. Then symmetrize and normalize to sum to one: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$
3. Low-dimensional similarities:

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = k\left(\left\|\mathbf{y}_i - \mathbf{y}_j\right\|\right), \quad Z = \sum_{k \neq l} w_{kl}$$

4. SNE: $k(d) = \exp\left(-d^2\right)$ and t-SNE: $k(d) = 1/\left(1 + d^2\right)$

t-Stochastic Neighbourhood Embedding (t-SNE)

t-SNE Algorithm

1. High-dimensional similarities:

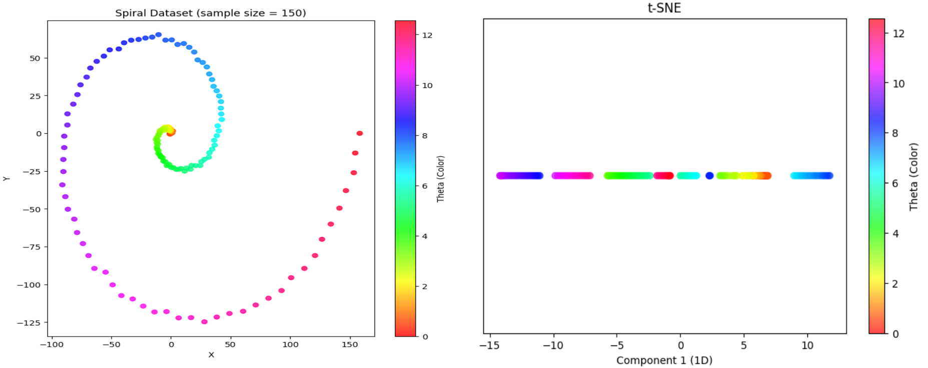
$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma_i^2\right)}$$

2. Then symmetrize and normalize to sum to one: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$
3. Low-dimensional similarities:

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = k\left(\left\|\mathbf{y}_i - \mathbf{y}_j\right\|\right), \quad Z = \sum_{k \neq l} w_{kl}$$

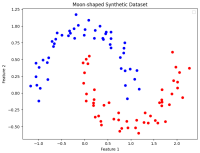
4. SNE: $k(d) = \exp\left(-d^2\right)$ and t-SNE: $k(d) = 1/\left(1 + d^2\right)$
5. $\mathcal{L} = -\sum_{i,j} p_{ij} \log q_{ij} = -\sum_{i,j} p_{ij} \log w_{ij} + \log \sum_{i,j} w_{ij}$
6. Apply gradient descent

t-Stochastic Neighbourhood Embedding (t-SNE)



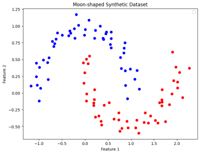
Comparison among all methods (discussed here)

2-d Data (Half Moons Data)

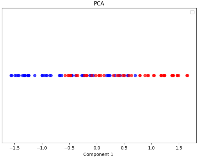


(a) Actual data

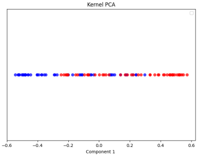
2-d Data (Half Moons Data)



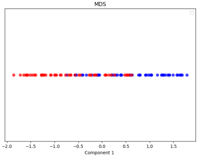
(a) Actual data



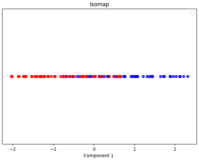
(b) PCA



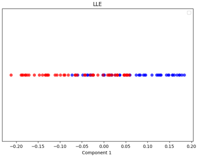
(c) KPCA



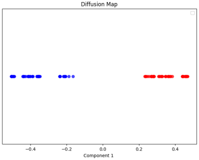
(d) MDS



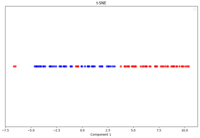
(e) ISOMAP



(f) LLE



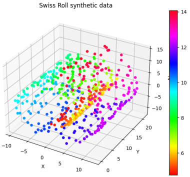
(g) Diffusion Map



(h) t-SNE

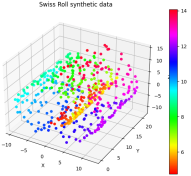
Figure 14: Comparison among six methods for half moon data ($N = 100$)

3-d Data (Swiss Roll)

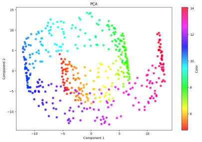


(a) Actual data

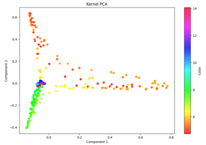
3-d Data (Swiss Roll)



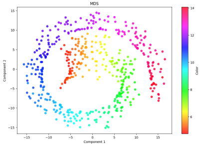
(a) Actual data



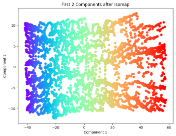
(b) PCA



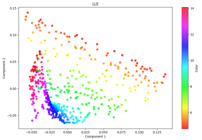
(c) KPCA



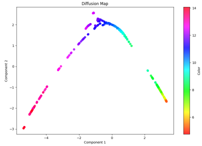
(d) MDS



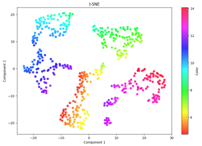
(e) ISOMAP



(f) LLE



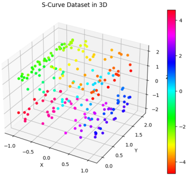
(g) Diffusion Map



(h) t-SNE

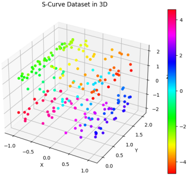
Figure 15: Comparison among six methods for swiss roll data ($N = 500$)

3-d Data (S-Curve Data)

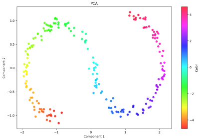


(a) Actual data

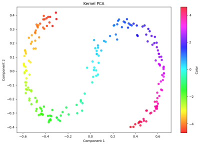
3-d Data (S-Curve Data)



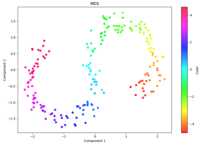
(a) Actual data



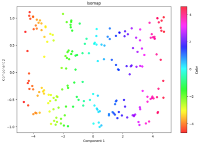
(b) PCA



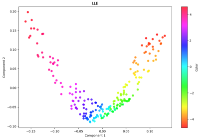
(c) KPCA



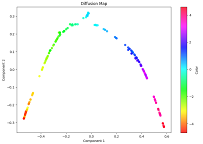
(d) MDS



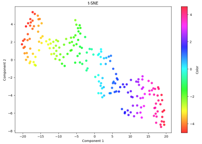
(e) ISOMAP



(f) LLE



(g) Diffusion Map



(h) t-SNE

Figure 16: Comparison among six methods for S-curve data ($N = 200$)

Challenges on applying on Real-life Data

Where is the problem?

Imagine a library with bookshelves (features) with two scenarios:

- Many shelves with few books (low spread-outness).
- Few shelves with books scattered across (high spread-outness).

Traditional methods (e.g., number of features) can't capture this "spread-outness." **Here's where fractal dimension comes in.**

Definition: q-Dimension

- Fractal dimension (DF) refers to dimensions of fractals (capacity, correlation, information). q-dimension unifies these.
- Suppose $\mathbf{y}$ is a random variable with DF $F(.)$ and pdf $f(.)$
- For $\epsilon > 0$, support of F is covered with a grid of cubes with edge length ϵ
- $N(\epsilon)$ be the number of cubes intersecting the support and p_i the probability of populated cubes:

$$D_q = \lim_{\epsilon \to 0} \frac{\log \left(\sum_{i=1}^{N(\epsilon)} p_i^q \right)}{(q-1)\log(\epsilon)}$$

- If the limit exists, D_q is the q-dimension of F.

Capacity Dimension

- Setting $q = 0$ in the q-dimension formula yields the capacity dimension (d_{cap}).
- Focuses on the number of covering boxes ($N(\epsilon)$) as cube size (ϵ) shrinks.

$$d_{cap} = \lim_{\epsilon \to 0} \frac{\log(N(\epsilon))}{\log(\epsilon)}$$

- Unlike other dimensions, it ignores individual point probabilities.

Application on a Synthetic Dataset

1. Generation n=500 observation from N(0, I_p) where I_p is a Identity matrix of order p=1000 and it is normalised.
2. Box Counting dimension is calculated by the above algorithm

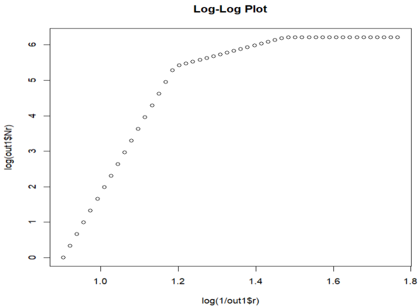


Figure 17: Log-Log Plot

3. Box Counting Dimension Estimated: 16.77263

Application on Yale Dataset

Application on Yale Dataset

Intrinsic dimension was found out to be 12

Intrinsic dimension was found out to be 12

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

Further Exploration

Further Exploration

- Computational Complexity
- Estimation of Intrinsic Dimension

Further Exploration

- Computational Complexity
- Estimation of Intrinsic Dimension
- Use of non-linear modelling architecture after linear dimension reduction over Nonlinear dimension reduction?

Further Exploration

- Computational Complexity
- Estimation of Intrinsic Dimension
- Use of non-linear modelling architecture after linear dimension reduction over Nonlinear dimension reduction?
- Extensions of the NLDR methods to incorporate handling out-of-sample data

References

S. Bernhard, S. Alexander, and M. Klaus-Robert. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10, 1998.

V. de Silva and J. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. 2002.

B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley. Locally linear embedding and its variants: Tutorial and survey. 2020.

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

C. Williams. On a connection between kernel pca and metric multidimensional scaling. 2002.