

# Lecture 14

## Nonlinear PCA and Manifold Learning

Reading: Izenman 2008: Chapter 16.1-16.2; 16.5-16.6

*DSA 8070 Multivariate Analysis*

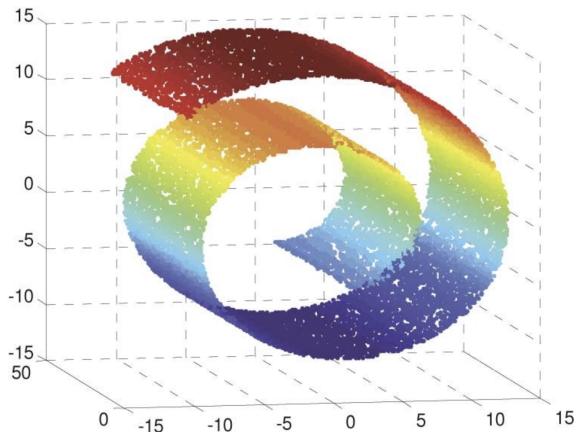
Whitney Huang  
Clemson University

## 1 Nonlinear PCA

- High-dimensional data often lie on low-dimensional manifolds
- Linear methods like PCA may fail to capture intrinsic geometry
- Goal: find lower-dimensional embedding that preserves structure

## Swiss Roll Example: Motivation for Manifold Learning

- A classic synthetic example for nonlinear dimensionality reduction.
- Data lie on a 2-dimensional manifold smoothly embedded in  $\mathbb{R}^3$ .
- Structure: a flat rectangle “rolled up” into a spiral surface.



By the end of this lecture, you will be able to:

- Understand the motivation behind manifold learning

By the end of this lecture, you will be able to:

- Understand the motivation behind manifold learning
- Describe key nonlinear dimensionality reduction techniques

By the end of this lecture, you will be able to:

- Understand the motivation behind manifold learning
- Describe key nonlinear dimensionality reduction techniques
- Implement and interpret, **Nonlinear PCA methods**, **Isomap**, **LLE**, and **t-SNE** in  $\mathbb{R}$

By the end of this lecture, you will be able to:

- Understand the motivation behind manifold learning
- Describe key nonlinear dimensionality reduction techniques
- Implement and interpret, [Nonlinear PCA methods](#), [Isomap](#), [LLE](#), and [t-SNE](#) in  $\mathbb{R}$
- Compare manifold methods to [Linear PCA](#)



## Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .

## Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:

## Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)

## Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)

## Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)
- Idea: generalize PCA so that we still

# Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)
- Idea: generalize PCA so that we still
  - Reduce dimension

# Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)
- Idea: generalize PCA so that we still
  - Reduce dimension
  - Capture major modes of variation

# Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)
- Idea: generalize PCA so that we still
  - Reduce dimension
  - Capture major modes of variation
  - Allow **nonlinear** transformations of the original variables



# Why Nonlinear PCA?

- Classical PCA assumes that the main structure of the data lies in a **linear subspace** of  $\mathbb{R}^p$ .
- For many data sets, structure is **nonlinear**:
  - Curved manifolds (e.g., Swiss roll)
  - Nonlinear functional relationships (e.g., quadratic curves)
- Idea: generalize PCA so that we still
  - Reduce dimension
  - Capture major modes of variation
  - Allow **nonlinear** transformations of the original variables
- There is no single unique “nonlinear PCA” – different methods generalize different properties of PCA

## Polynomial PCA: Basic Idea

- Suppose  $X \in \mathbb{R}^p$  is the original data vector

## Polynomial PCA: Basic Idea

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1X_2, \dots, X_{p-1}X_p)$$

## Polynomial PCA: Basic Idea

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1X_2, \dots, X_{p-1}X_p)$$

- Apply **ordinary PCA** to the expanded data  $X'$ :

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1X_2, \dots, X_{p-1}X_p)$$

- Apply **ordinary PCA** to the expanded data  $X'$ :
  - Center  $X'$

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1X_2, \dots, X_{p-1}X_p)$$

- Apply **ordinary PCA** to the expanded data  $X'$ :
  - Center  $X'$
  - Compute covariance matrix of  $X'$

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1 X_2, \dots, X_{p-1} X_p)$$

- Apply **ordinary PCA** to the expanded data  $X'$ :
  - Center  $X'$
  - Compute covariance matrix of  $X'$
  - Eigenvalues/eigenvectors give “principal components” in the polynomial feature space

- Suppose  $X \in \mathbb{R}^p$  is the original data vector
- Construct an **expanded feature vector**  $X'$  by adding polynomial terms, e.g. for quadratic PCA:

$$X' = (X_1, \dots, X_p, X_1^2, \dots, X_p^2, X_1 X_2, \dots, X_{p-1} X_p)$$

- Apply **ordinary PCA** to the expanded data  $X'$ :
  - Center  $X'$
  - Compute covariance matrix of  $X'$
  - Eigenvalues/eigenvectors give “principal components” in the polynomial feature space
- If the true relationship is approximately polynomial, small eigenvalues can correspond to **approximate nonlinear constraints**



- Advantages:

- Advantages:
  - Simple conceptually: just augment features and run PCA

- Advantages:
  - Simple conceptually: just augment features and run PCA
  - Can capture simple nonlinear structure (quadratic, cubic, etc.)

- Advantages:
  - Simple conceptually: just augment features and run PCA
  - Can capture simple nonlinear structure (quadratic, cubic, etc.)
- Disadvantages:

- Advantages:
  - Simple conceptually: just augment features and run PCA
  - Can capture simple nonlinear structure (quadratic, cubic, etc.)
- Disadvantages:
  - Dimension of  $X'$  grows quickly with polynomial degree  $d$  and number of variables  $p$

- Advantages:
  - Simple conceptually: just augment features and run PCA
  - Can capture simple nonlinear structure (quadratic, cubic, etc.)
- Disadvantages:
  - Dimension of  $X'$  grows quickly with polynomial degree  $d$  and number of variables  $p$
  - Risk of overfitting and numerical instability

- Instead of explicitly constructing polynomial (or other) features, map  $x \in \mathbb{R}^p$  to a feature space  $H$ :

$$\Phi : \mathbb{R}^p \rightarrow H,$$

and perform **linear PCA** on  $\Phi(x_1), \dots, \Phi(x_n)$

- Instead of explicitly constructing polynomial (or other) features, map  $x \in \mathbb{R}^p$  to a feature space  $H$ :

$$\Phi : \mathbb{R}^p \rightarrow H,$$

and perform **linear PCA** on  $\Phi(x_1), \dots, \Phi(x_n)$

- We never need  $\Phi(x)$  explicitly, only inner products

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_H$$



- Instead of explicitly constructing polynomial (or other) features, map  $x \in \mathbb{R}^p$  to a feature space  $H$ :

$$\Phi : \mathbb{R}^p \rightarrow H,$$

and perform **linear PCA** on  $\Phi(x_1), \dots, \Phi(x_n)$

- We never need  $\Phi(x)$  explicitly, only inner products

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_H$$

- A **kernel**  $K(x_i, x_j)$  replaces the explicit feature map

- Instead of explicitly constructing polynomial (or other) features, map  $x \in \mathbb{R}^p$  to a feature space  $H$ :

$$\Phi : \mathbb{R}^p \rightarrow H,$$

and perform **linear PCA** on  $\Phi(x_1), \dots, \Phi(x_n)$

- We never need  $\Phi(x)$  explicitly, only inner products

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_H$$

- A **kernel**  $K(x_i, x_j)$  replaces the explicit feature map
- Kernel PCA = PCA in feature space defined by  $K$

- 1 Choose a kernel  $K$ , e.g.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (\text{Gaussian / RBF kernel}).$$

- 1 Choose a kernel  $K$ , e.g.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (\text{Gaussian / RBF kernel}).$$

- 2 Form the  $n \times n$  Gram matrix  $K_{ij} = K(x_i, x_j)$

- 1 Choose a kernel  $K$ , e.g.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (\text{Gaussian / RBF kernel}).$$

- 2 Form the  $n \times n$  Gram matrix  $K_{ij} = K(x_i, x_j)$

- 3 Center  $K$  in feature space:

$$\tilde{K} = K - \mathbf{1}K/n - K\mathbf{1}/n + \mathbf{1}K\mathbf{1}/n^2,$$

where  $\mathbf{1}$  is the all-ones matrix

- 1 Choose a kernel  $K$ , e.g.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (\text{Gaussian / RBF kernel}).$$

- 2 Form the  $n \times n$  Gram matrix  $K_{ij} = K(x_i, x_j)$

- 3 Center  $K$  in feature space:

$$\tilde{K} = K - \mathbf{1}K/n - K\mathbf{1}/n + \mathbf{1}K\mathbf{1}/n^2,$$

where  $\mathbf{1}$  is the all-ones matrix

- 4 Compute eigenvalues/eigenvectors of  $\tilde{K}$ :

$$\tilde{K}\alpha^{(\ell)} = n\lambda_\ell\alpha^{(\ell)}$$

- 1 Choose a kernel  $K$ , e.g.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (\text{Gaussian / RBF kernel}).$$

- 2 Form the  $n \times n$  Gram matrix  $K_{ij} = K(x_i, x_j)$

- 3 Center  $K$  in feature space:

$$\tilde{K} = K - \mathbf{1}K/n - K\mathbf{1}/n + \mathbf{1}K\mathbf{1}/n^2,$$

where  $\mathbf{1}$  is the all-ones matrix

- 4 Compute eigenvalues/eigenvectors of  $\tilde{K}$ :

$$\tilde{K}\alpha^{(\ell)} = n\lambda_\ell\alpha^{(\ell)}$$

- 5 The coordinates of data point  $x_i$  on kernel PC  $\ell$  are proportional to the  $i$ th component of  $\alpha^{(\ell)}$

- Kernel PCA finds directions of maximal variance in a **nonlinear feature space**.
- For suitable kernels, kernel PCA can:
  - capture curved manifolds,
  - partially “unfold” structures like the Swiss roll,
  - behave similarly to metric MDS based on a kernel-induced distance.
- The kernel choice (e.g., bandwidth  $\sigma$  in RBF kernel) controls the balance between:
  - local vs. global structure,
  - smooth vs. noisy embeddings.



Compare 2D linear PCA and RBF kernel PCA on the same Swiss roll dataset (coloured by height  $h$ )

### Linear PCA

- Projects data onto a best linear 2D subspace
- Swiss roll remains curved and “folded”
- Colour gradient (in  $h$ ) is not monotone along any axis

### Kernel PCA (RBF)

- Applies PCA in a nonlinear feature space defined by the kernel
- Effectively unrolls the manifold into 2D
- Colour gradient in  $h$  varies smoothly across the embedding

**Key message:** Linear PCA is limited to global linear structure, whereas kernel PCA can recover nonlinear manifold structure when the kernel is chosen appropriately

- Linear PCA can fail when the data lie on a nonlinear manifold
- Polynomial PCA:
  - PCA on explicitly expanded polynomial features
  - Simple but can be high-dimensional
- Kernel PCA:
  - PCA in an implicit feature space defined by a kernel
  - More flexible and scalable than explicit polynomial expansion

- Data points lie on a nonlinear subspace (manifold)
- Manifold learning tries to “unroll” this space
- Preserves local or global geometric relationships

- Combines classical MDS with geodesic distance
- Steps:
  - 1 Construct neighborhood graph (k-nearest or epsilon)
  - 2 Compute shortest path distances (Floyd-Warshall)
  - 3 Apply MDS on the geodesic distance matrix
- Captures global geometry

- Preserves local linear relationships between neighbors
- Steps:
  - 1 Identify k-nearest neighbors
  - 2 Compute weights to reconstruct each point from neighbors
  - 3 Find embedding that preserves weights
- Sensitive to noise and parameter choice

## t-SNE (t-distributed Stochastic Neighbor Embedding)

- Probabilistic approach to preserve pairwise similarities
- Converts distances to probabilities in high and low dimensions
- Optimizes Kullback-Leibler divergence between distributions
- Good for visualization (not true metric embedding)

# UMAP (Uniform Manifold Approximation and Projection)

- Balances local and global structure preservation
- Faster than t-SNE and scales better
- Based on fuzzy simplicial sets and topological data analysis

- PCA is linear and global
- Manifold methods are nonlinear, and often local
- Use PCA when interpretability and simplicity matter
- Use t-SNE, Isomap, or UMAP for visualizing structure



- Polynomial PCA and kernel PCA:
  - Extend PCA using richer feature spaces
  - Remain **variance-based** methods
- Methods like Isomap, LLE:
  - Focus on preserving **geodesic distances** or **local neighborhoods**
  - Explicitly target manifold geometry

- Manifold learning methods uncover low-dimensional structures
- Isomap and LLE preserve distances or local geometry
- t-SNE and UMAP are widely used for high-dimensional data visualization