

INT104 ARTIFICIAL INTELLIGENCE

LECTURE 3- DIMENSIONALITY REDUCTION

Sichen Liu

Sichen.Liu@xjtlu.edu.cn



Xi'an Jiaotong-Liverpool University

西交利物浦大學



CONTENT

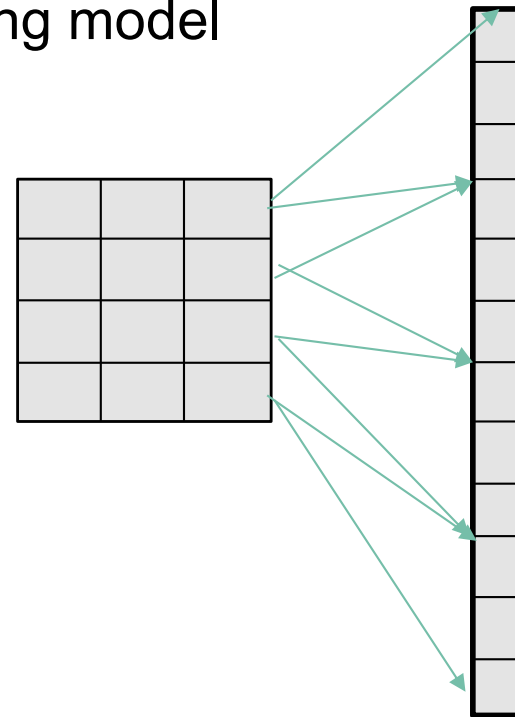
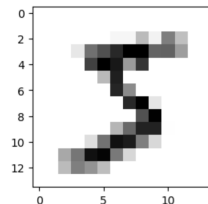
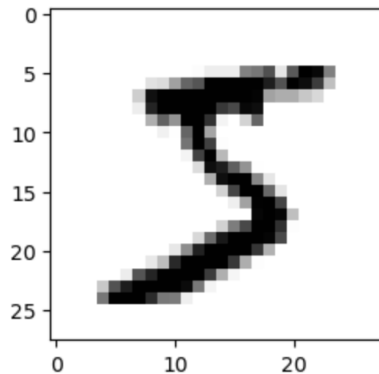
- Why need Dimensionality Reduction
- Principal Component Analysis (PCA)
- Locally Linear Embedding (LLE)
- Other Dimensionality Reduction Techniques



Dimensionality Reduction

Data with high dimensions:

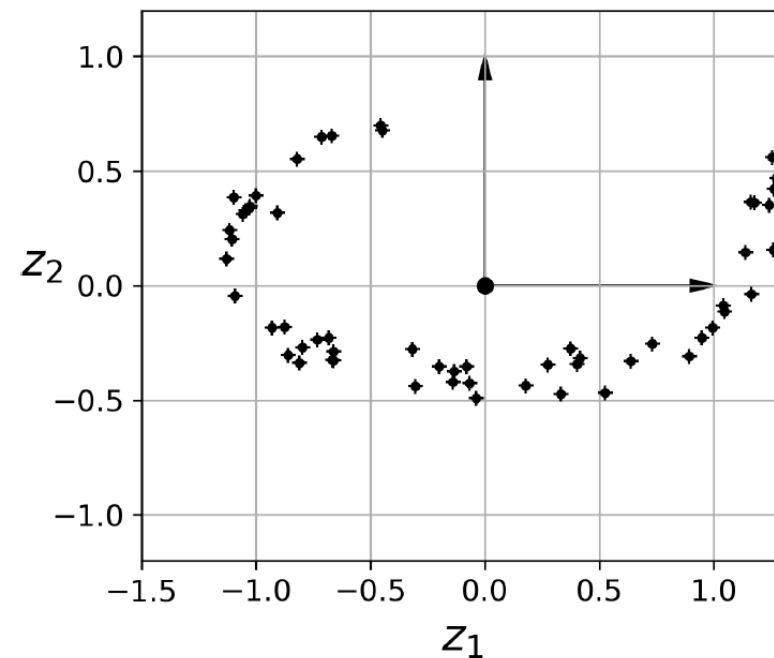
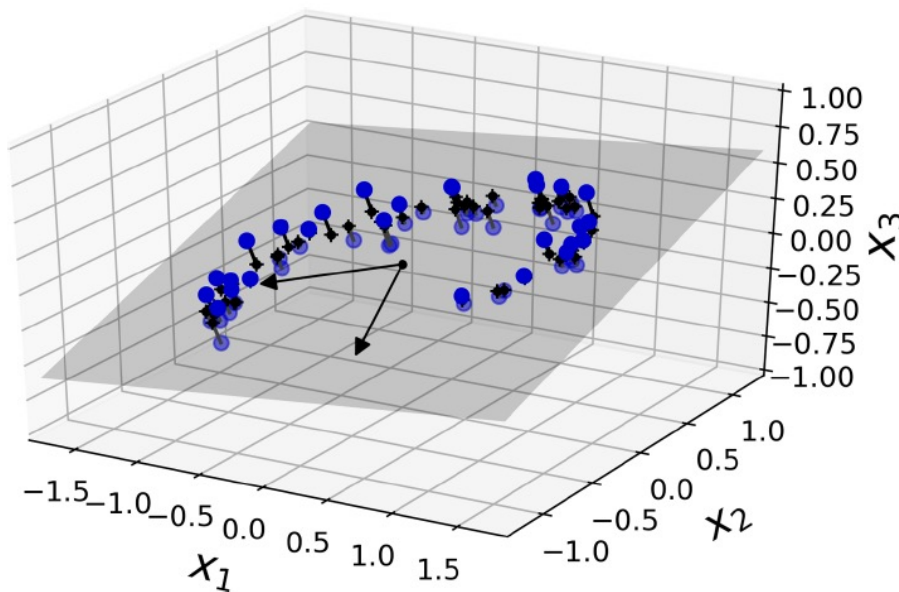
- High computational complexity
- May contain many irrelevant or redundant features
- Difficulty in visualization
- With high risk of getting an overfitting model



Approaches for Dimensionality Reduction

Projection:

- Data is not spread out uniformly across all dimensions. (All the data lies within (or close to) a much lower-dimensional subspace of the high-dimensional space.)



Principal Component Analysis (PCA)

Preserving the Variance:

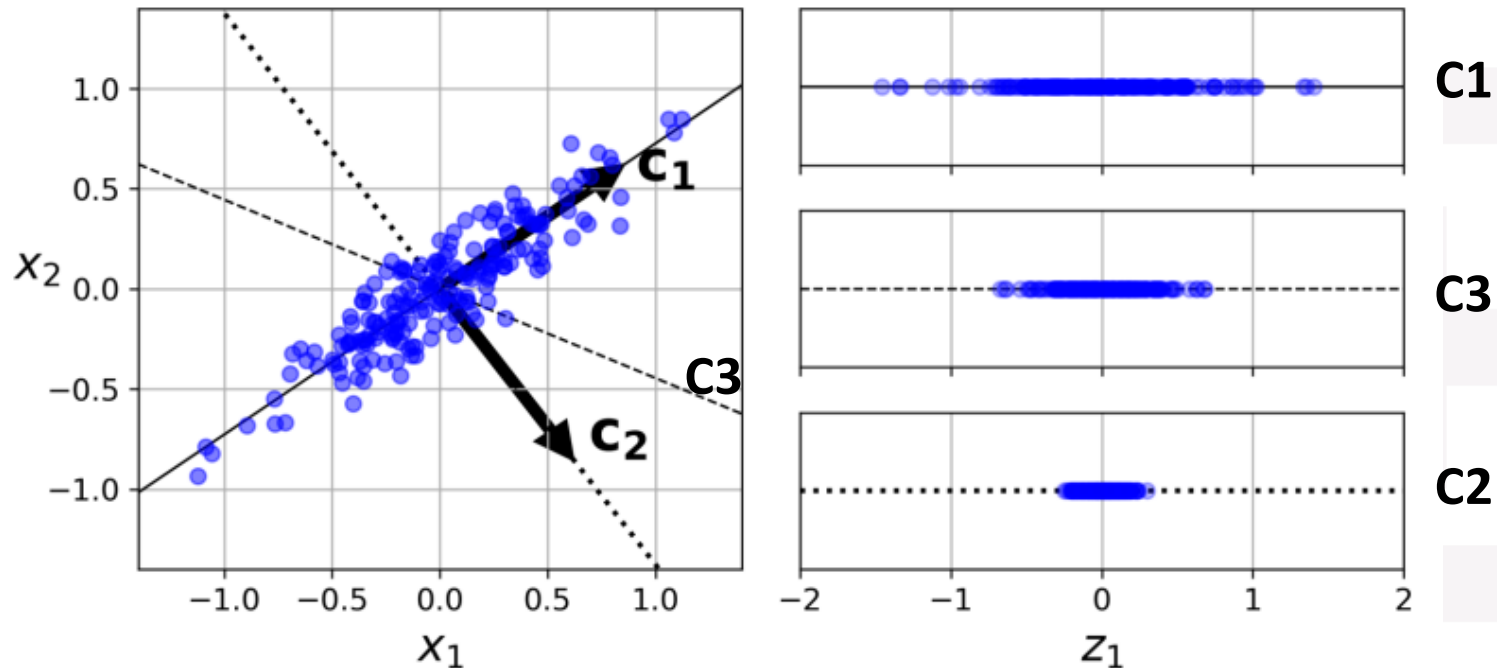


Figure 8-7. Selecting the subspace to project on

PCA identifies the axis that accounts for the largest amount of variance in the training set.



Principal Component Analysis (PCA)

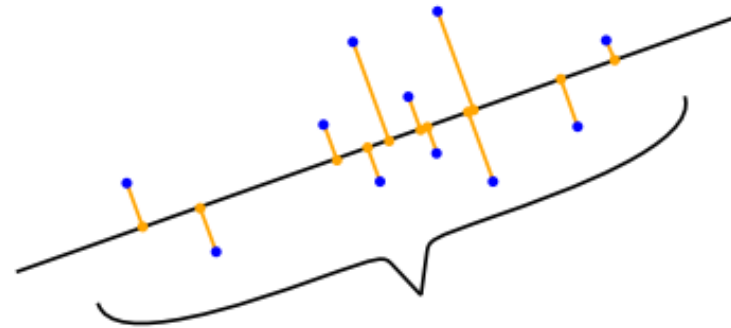
-Variance on C1

$$\begin{aligned} V_1 &= \frac{1}{M} \sum_{i=1}^M (\mathbf{c}_1^\top \mathbf{x}^{(i)})^2 = \frac{1}{M} \sum_{i=1}^M \mathbf{c}_1^\top \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \mathbf{c}_1 = \mathbf{c}_1^\top \left(\frac{1}{M} \sum_{i=1}^M \mathbf{x}^{(i)} \mathbf{x}^{(i)\top} \right) \mathbf{c}_1 \\ &= \mathbf{c}_1^\top \mathbf{S} \mathbf{c}_1 \end{aligned}$$

-Data covariance matrix

$$\mathbf{S} = \frac{1}{M} \sum_{i=1}^M \mathbf{x}^{(i)} \mathbf{x}^{(i)\top}$$

- S is an N*N matrix, N is the number of features, M is the total number of data points.



Principal Component Analysis (PCA)

-Constrained optimization problem

$$\begin{aligned} \max_{\mathbf{c}_1} \quad & \mathbf{c}_1^\top S \mathbf{c}_1 \\ \text{subject to} \quad & \|\mathbf{c}_1\|^2 = 1 \end{aligned}$$

-Lagrange equation $\mathcal{L}(\mathbf{c}_1, \lambda_1) = \mathbf{c}_1^\top S \mathbf{c}_1 + \lambda_1(1 - \mathbf{c}_1^\top \mathbf{c}_1)$

-Solve this constrained optimization problem

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}_1} = 2S\mathbf{c}_1 - 2\lambda_1\mathbf{c}_1 \quad \frac{\partial \mathcal{L}}{\partial \lambda_1} = 1 - \mathbf{c}_1^\top \mathbf{c}_1$$

- Setting these partial derivatives to 0 gives us the relations:

$$S\mathbf{c}_1 = \lambda_1\mathbf{c}_1 \quad \text{and} \quad \mathbf{c}_1^\top \mathbf{c}_1 = 1$$

- Variance on C1

$$V_1 = \mathbf{c}_1^\top S \mathbf{c}_1 = \lambda_1 \mathbf{c}_1^\top \mathbf{c}_1 = \lambda_1$$



Practice: PCA

Given a dataset that consists of the following points below:

$A=(2, 3)$, $B=(5, 5)$, $C=(6, 6)$, $D=(8,9)$

1. Calculate the covariance matrix for the dataset.
2. Calculate the eigenvalues and eigenvectors of the covariance matrix.



PCA

Singular Value Decomposition (SVD)

Theorem: Let $A \in R^{m \times n}$ be a rectangular matrix of rank $r \in [0, \min(m, n)]$. The SVD of A is a decomposition of the form

$$\begin{matrix} n \\ \boxed{A} \\ m \end{matrix} = \begin{matrix} m \\ \boxed{U} \\ m \end{matrix} \begin{matrix} m \\ \boxed{\Sigma} \\ m \end{matrix} \begin{matrix} n \\ \boxed{V^T} \\ n \end{matrix}$$

- $U \in R^{m \times m}$ is an orthogonal matrix with column vectors $u_i, i = 1, \dots, m$,
- $V \in R^{n \times n}$ an orthogonal matrix with column vectors $v_j, j = 1, \dots, n$.
- Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0$ and $\Sigma_{ij} = 0, i \neq j$
- *The singular value matrix Σ is unique*



PCA

Singular Value Decomposition (SVD)

$$A = [x_1 \quad \dots \quad x_n]_{m \times n} = U \Sigma V^T = [u_1 \quad \dots \quad u_m]_{m \times m} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix}_{m \times n} [v_1 \quad \dots \quad v_n]^T_{n \times n}$$

V contains the unit vectors that define all the principal components that we are looking for.

```
X_centered = (X - X.mean(axis=0))/X.std(axis=0)
U, s, Vt = np.linalg.svd(X_centered)
c1 = Vt.T[:, 0]
c2 = Vt.T[:, 1]
```



Principal Component Analysis (PCA)

Principal components matrix

$$\mathbf{V} = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & \cdots & | \end{pmatrix}$$

$\mathbf{c}_1, \mathbf{c}_2 \dots \mathbf{c}_n$ are orthogonal

Projecting Down to d Dimension:

$$X_{d-proj} = XV_d$$

V_d is the first d eigen vectors of data covariance matrix

Explained Variance Ratio

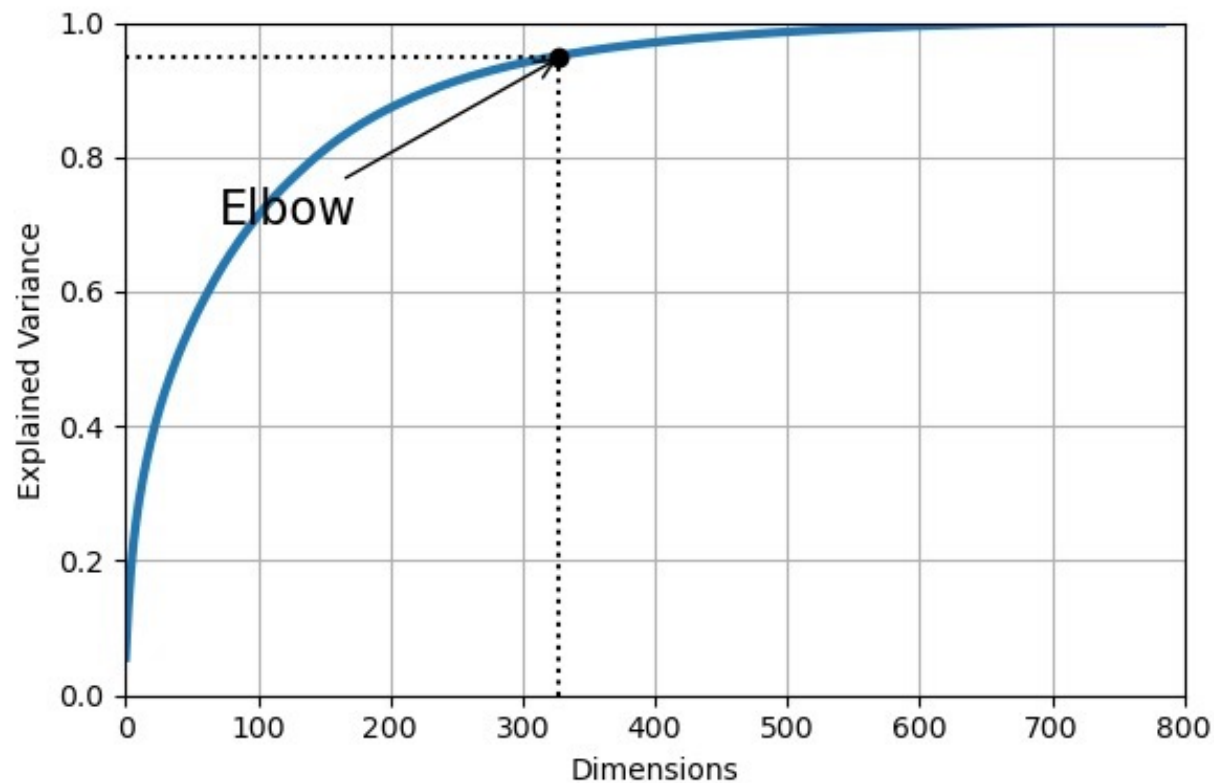
$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \dots + \lambda_n} \text{ (eigenvalue/ total eigenvalue)}$$



PCA

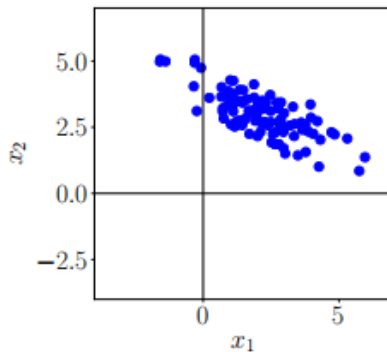
Choosing the Right Number of Dimensions:

- Choose the number of dimensions that add up to sufficiently large portion of the variance (e.g., 95%)
- $\frac{\lambda_1 + \dots + \lambda_d}{\lambda_1 + \lambda_2 + \dots + \lambda_n} > 95\%$

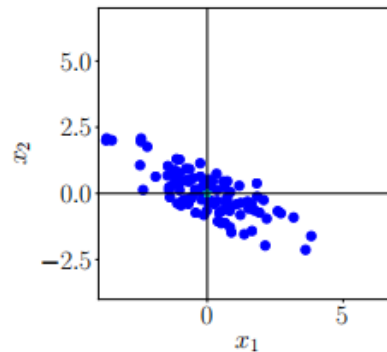


PCA

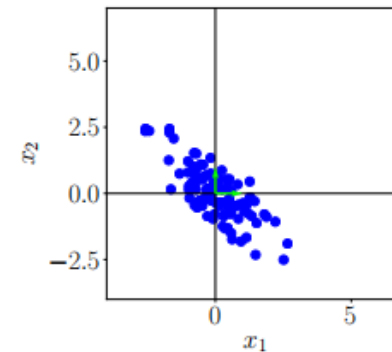
Key steps of PCA in practice



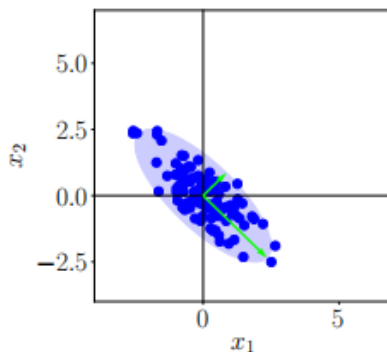
(a) Original dataset.



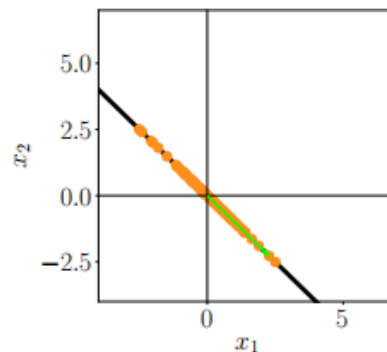
(b) Step 1: Centering by subtracting the mean from each data point.



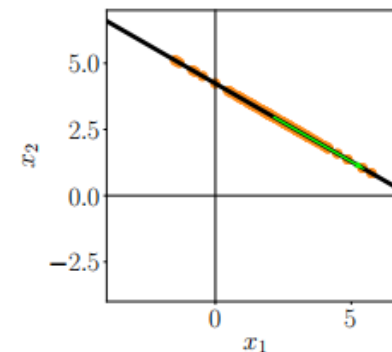
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).



(e) Step 4: Project data onto the principal subspace.



(f) Undo the standardization and move projected data back into the original data space from (a).



PCA

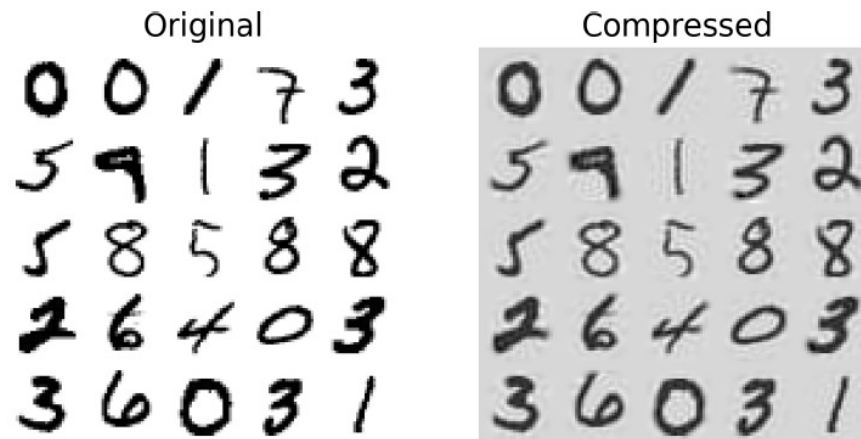
PCA for Compression

- Projecting Down to d Dimension

$$X_{d-proj} = XV_d$$

- PCA inverse transformation, back to the original number of dimensions

$$X_{recovered} = X_{d-proj}V_d^T$$



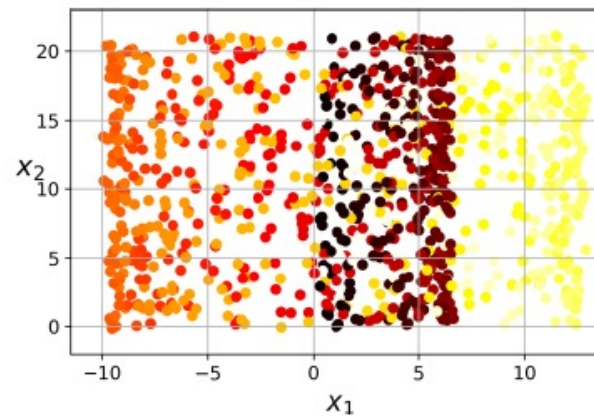
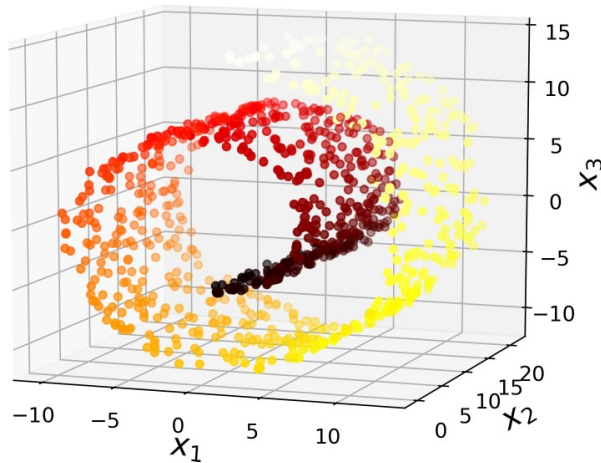
(Example MNIST data: 40% original size preserves 95% variance)



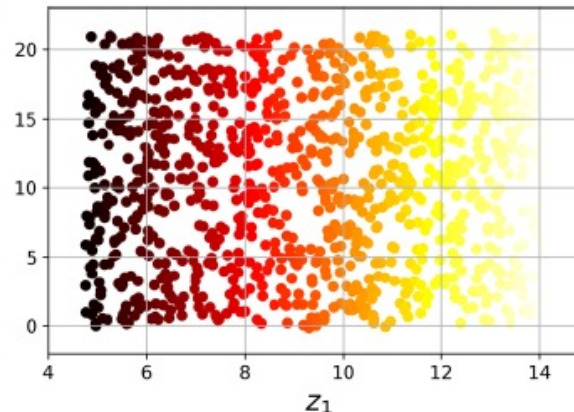
Approaches for Dimensionality Reduction

Manifold Learning

- Data lies on d-dimensional manifold is a part of an n-dimensional space (where $d < n$)



*Simply projecting
onto a plane*



*Unrolling the Swiss
roll*



Locally Linear Embedding (LLE)

LLE is a powerful *nonlinear dimensionality reduction* (NLDR) technique. It is a Manifold Learning technique that does not rely on projections

Step one: Linearly modeling local relationships

$$\begin{aligned} \widehat{\mathbf{W}} = \operatorname{argmin}_{\mathbf{W}} \sum_{i=1}^m \left(\mathbf{x}^{(i)} - \sum_{j=1}^m w_{i,j} \mathbf{x}^{(j)} \right)^2 \\ \text{subject to } \begin{cases} w_{i,j} = 0 & \text{if } \mathbf{x}^{(j)} \text{ is not one of the } k \text{ c.n. of } \mathbf{x}^{(i)} \\ \sum_{j=1}^m w_{i,j} = 1 & \text{for } i = 1, 2, \dots, m \end{cases} \end{aligned}$$

Step two: Reducing dimensionality while preserving relationships

$$\widehat{\mathbf{Z}} = \operatorname{argmin}_{\mathbf{Z}} \sum_{i=1}^m \left(\mathbf{z}^{(i)} - \sum_{j=1}^m \widehat{w}_{i,j} \mathbf{z}^{(j)} \right)^2$$



Locally Linear Embedding (LLE)

```
from sklearn.manifold import LocallyLinearEmbedding  
  
lle = LocallyLinearEmbedding(n_components=2, n_neighbors=10)  
X_reduced = lle.fit_transform(X)
```

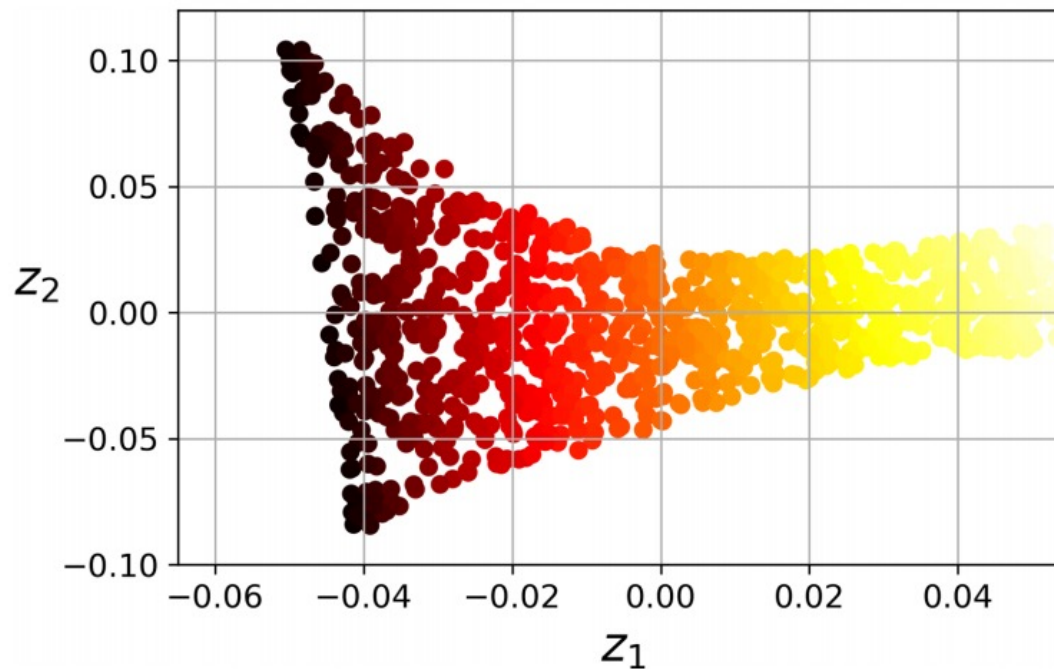
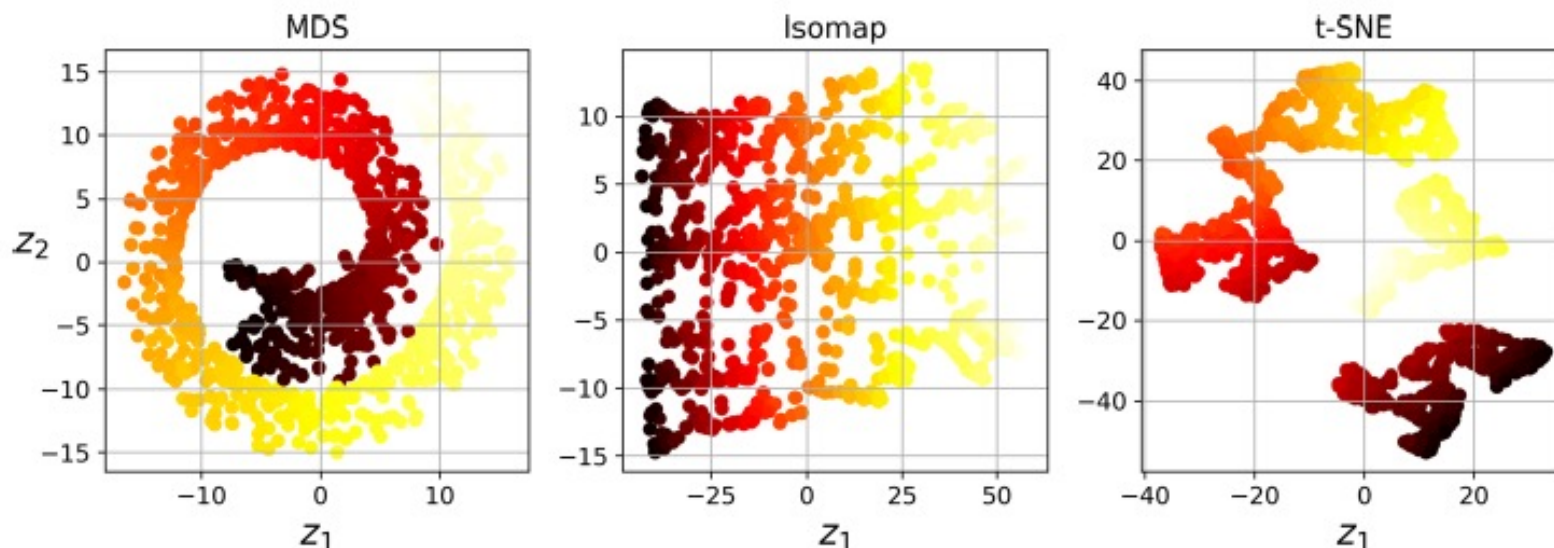


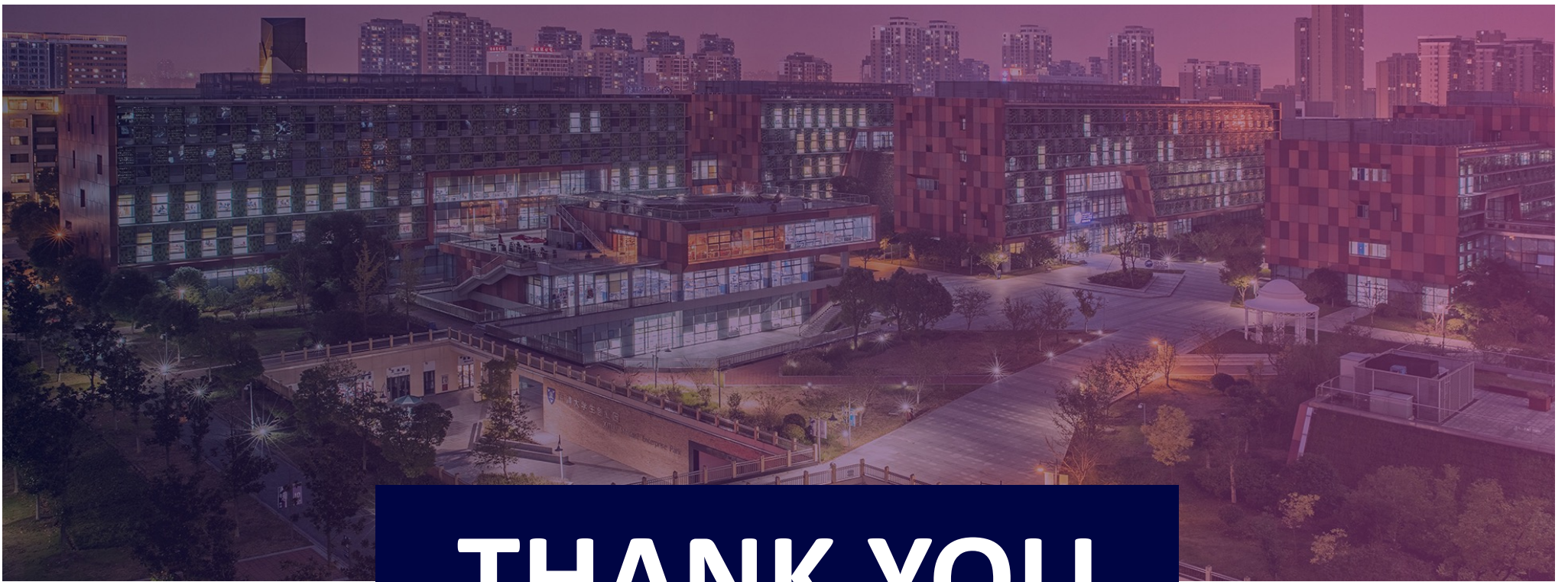
Figure 8-12. Unrolled Swiss roll using LLE



Other Techniques

- Multidimensional Scaling (MDS)
Trying to preserve the distances between the instances.
- Isomap
Trying to preserve the geodesic distances between the instances.
- t-Distributed Stochastic Neighbor Embedding (t-SNE)
Trying to keep similar instances close and dissimilar instances apart.





THANK YOU



Xi'an Jiaotong-Liverpool University
西交利物浦大學

