Report on Principal Component Analysis (PCA)

xiong rui

October 18, 2025

Contents

1 Mathematical Symbols

Table 1: Summary of Mathematical Notations

No.	Notation	Definition
1	X	$m \times n$ data matrix (m: measurement types, n: samples); each column is a single sample $\vec{X}$.
2	$\vec{X}$	Column vector of X; represents one experimental sample (e.g., 6D position from 3 cameras).
3	C_X	Covariance matrix of X; for centered X_{cent}, $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$ ($m \times m$ symmetric matrix).
4	P	Orthonormal basis matrix for PCA ($m \times m$); rows are principal components (PCs), satisfying $PP^T = I$.
5	p_i	i-th principal component; i-th row of P ($p_i \in \mathbb{R}^m$), eigenvector of C_X with $p_i^T p_j = \delta_{ij}$.
6	Y	Transformed data via PCA; $Y = PX_{\text{cent}}$ ($m \times n$), rows = projections of X_{cent} onto PCs.
7	Y_k	k-dimensional reduced data; $Y_k = P_k^T X_{\text{cent}}$ ($k \times n$), P_k = first k columns of V (SVD matrix).
8	σ_A^2	Variance of zero-mean variable $A = \{a_1, ..., a_n\}$; $\sigma_A^2 = \frac{1}{n} \sum_{i=1}^n a_i^2 = \text{Cov}(A, A)$.
9	σ_{signal}^2	Signal variance; variance from true data structure (e.g., spring motion), numerator of SNR.
10	σ_{noise}^2	Noise variance; variance from measurement error (e.g., camera noise), denominator of SNR.
11	$\text{Cov}(A, B)$	Covariance of A, B; $\text{Cov}(A, B) = \frac{1}{n} \sum_{i=1}^n a_i b_i$, quantifies linear correlation.
12	SNR	Signal-to-Noise Ratio; $\text{SNR} = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{noise}}^2}$, $\text{SNR} \gg 1$ means strong signal.
13	λ_i	Eigenvalue of C_X; equals variance of X_{cent} on p_i, sorted as $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$.
14	E	Eigenvector matrix for C_X; $C_X = EDE^T$ (E: $m \times m$ orthogonal, columns = eigenvectors of C_X).
15	D	Eigenvalue matrix for C_X; diagonal matrix with $\lambda_1, ..., \lambda_m$ as diagonal entries.
16	σ_i	Singular value of X_{cent}; diagonal entry of Σ (SVD), satisfies $\sigma_i = \sqrt{n\lambda_i}$.
17	Σ	Singular value matrix in SVD; $X_{\text{cent}} = U\Sigma V^T$ ($m \times n$ diagonal), entries = $\sigma_1 \geq ... \geq \sigma_r$ (r = rank).
18	U, V	Orthogonal matrices in SVD; $UU^T = I$ ($m \times m$), $VV^T = I$ ($n \times n$); columns of V = PCs of X.
19	δ_{ij}	Kronecker delta; $\delta_{ij} = 1$ if $i = j$, else 0; describes orthogonality of PCs.
20	$\| \cdot \|_F$	Frobenius norm; $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$; used to define PCA reconstruction error.
21	$\text{tr}(\cdot)$	Trace of square matrix; sum of diagonal entries; $\text{tr}(C_X) = \sum_{i=1}^m \lambda_i$ (total variance).
22	X_{cent}	Centered data matrix; $X_{\text{cent}} = X - \mu_X \cdot \mathbb{1}^T$ ($m \times n$), where each row of X has zero mean.
23	Y_{SVD}	SVD auxiliary matrix; $Y_{\text{SVD}} = \frac{1}{\sqrt{n}} X_{\text{cent}}^T$ ($n \times m$), satisfying $Y_{\text{SVD}}^T Y_{\text{SVD}} = C_X$ (links SVD to covariance matrix).

2 Answers to Q&A

2.1 The Essence of Matrix

From a linear algebra perspective [Sec. III.B], a matrix $A \in \mathbb{R}^{m \times n}$ is a representation of linear transformation from vector space $\mathbb{R}^n$ to $\mathbb{R}^m$. It satisfies the linearity property: for any $\vec{x}, \vec{y} \in \mathbb{R}^n$ and scalars $a, b \in \mathbb{R}$,

$$A(a\vec{x} + b\vec{y}) = aA\vec{x} + bA\vec{y}.$$

In PCA, the orthonormal matrix $P \in \mathbb{R}^{m \times m}$ acts as a basis transformation tool: it maps data X from the original measurement basis (e.g., camera axes) to the PCA basis (principal components, PCs) via $Y = PX$, where Y denotes the transformed data.

2.2 Problems Addressed by PCA (According to Fig. 1)

Fig. 1 describes a spring-ball system: the ball oscillates along the 1D x-axis, but 3 cameras record 6D data (2D projections per camera) with noise. PCA addresses three core problems:

1. **High dimensionality**: (Implies PCA) Reduce 6D data to 1D (the true motion direction x) without losing key information.

2. **Redundancy**: Cameras record correlated data (e.g., x_A and x_B for nearby cameras), leading to $|\text{Cov}(x_A, x_B)| \gg 0$; PCA eliminates redundancy by diagonalizing the covariance matrix C_X.

3. **Noise contamination**: Noise introduces small random variance; PCA retains directions with large variance (high SNR) to filter out noise.

2.3 Assumptions and Limits of PCA (Mathematical Explanation for Sec. E)

2.3.1 Assumptions (Mathematical Formulation)

1. **Linearity**: PCA uses a linear basis transformation. For original data X and transformed data Y, $Y = PX$, where P is an orthonormal matrix (no nonlinear terms like PX^2 are allowed).

2. **Large variance implies important structure**: For the i-th PC p_i, its associated eigenvalue $\lambda_i = \text{Var}(p_i^T X)$ (eigenvalue of C_X) is proportional to signal strength. For useful components, the signal-to-noise ratio satisfies:
$$\text{SNR}_i = \frac{\lambda_i - \sigma_{\text{noise}}^2}{\sigma_{\text{noise}}^2} \gg 1,$$
where σ_{noise}^2 denotes noise variance.

3. **Orthonormality of PCs**: PCs satisfy $p_i^T p_j = \delta_{ij}$, meaning any two distinct PCs are orthogonal.

2.3.2 Limits

1. **Nonlinear data**: For data with nonlinear structure (e.g., ferris wheel motion: $x = \cos\theta, y = \sin\theta$), linear transformation $Y = PX$ cannot recover the true 1D variable θ.

2. **Non-Gaussian data**: If data is non-Gaussian (e.g., Fig. 6b), second-order statistics (covariance) fail to capture higher-order dependencies, leading to suboptimal dimensionality reduction.

2.4 Basis in Linear Algebra (Intuitive Explanation)

2.4.1 Intuitive Definition

A basis is a set of "orthogonal axes" that span a vector space without redundancy—intuitively, it is the minimal set of vectors needed to "cover" all points in the space.

2.4.2 Mathematical Definition

Let $V \subseteq \mathbb{R}^m$ be a vector subspace. A set $\mathcal{B} = \{\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_k\} \subseteq V$ is a basis of V if:

1. **Linear independence**: $\sum_{i=1}^{k} a_i \vec{b}_i = \vec{0} \implies a_i = 0$ for all i (no vector in $\mathcal{B}$ can be expressed as a linear combination of others).

2. **Spanning** V: For any $\vec{v} \in V$, there exist scalars $a_1, \ldots, a_k$ such that $\vec{v} = \sum_{i=1}^{k} a_i \vec{b}_i$ (all vectors in V can be constructed from $\mathcal{B}$).

2.4.3 Example

The "naive basis" for camera A's 2D data is $\mathcal{B}_{\text{naive}} = \{(1,0)^T, (0,1)^T\}$, which corresponds to the camera's pixel axes.

2.5 What Covariance Represents and Its Relations to Redundancy and Noise

2.5.1 Meaning of Covariance

For zero-mean variable sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, covariance quantifies linear correlation between A and B. Mathematically:

$$\text{Cov}(A, B) = \frac{1}{n}\sum_{i=1}^{n} a_i b_i = \frac{1}{n} AB^T,$$

where $A = [a_1, \ldots, a_n]$ and $B = [b_1, \ldots, b_n]$ are row vectors. It satisfies:

- $\text{Cov}(A, B) > 0$: Positive correlation;

- $\text{Cov}(A, B) = 0$: No linear correlation;

- $\text{Cov}(A, B) < 0$: Negative correlation.

2.5.2 Relations

1. **Covariance Redundancy**: ~~According to *A tutorial on Principal Components Analysis*,~~ redundancy between two zero-mean variables A and B is quantified by the correlation coefficient:

$$\rho_{AB} = \frac{|\text{Cov}(A, B)|}{\sigma_A \sigma_B}$$

where $\sigma_A = \sqrt{\text{Var}(A)}$ (variance of A). $\rho_{AB} \approx 1$ means high redundancy. For example, if $A = x_A$ (position from camera A) and $B = 2x_A$ (scaled position of the same camera), their covariance is $\text{Cov}(A, B) = 2\sigma_A^2$, leading to $\rho_{AB} = 1$ (full redundancy).

2. **Covariance Noise**: As stated in *A tutorial on Principal Components Analysis (2014)*, noise adds random variance to data. The total covariance matrix of the data satisfies:

$$C_{X_{\text{total}}} = C_{X_{\text{signal}}} + C_{X_{\text{noise}}}$$

Since noise is uncorrelated with the signal (i.e., $\text{Cov}(\text{signal}, \text{noise}) = 0$), it only increases the off-diagonal elements of C_X (spurious correlation between measurements).

2.6 Definitions of SNR, Variance, Redundancy and Their Relations

2.6.1 Definitions

1. **Variance**: For a zero-mean variable A, variance measures the spread of data:

$$\sigma_A^2 = \text{Var}(A) = \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \text{Cov}(A, A).$$

2. **SNR (Signal-to-Noise Ratio)**: SNR (Signal-to-Noise Ratio) quantifies signal strength relative to noise:

$$\text{SNR} = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{noise}}^2},$$

where σ_{signal}^2 is signal variance and σ_{noise}^2 is noise variance.

3. **Redundancy**: For two variables A and B, redundancy measures overlapping information:

$$R(A, B) = \frac{|\text{Cov}(A, B)|}{\sigma_A \sigma_B}, \quad R(A, B) \in [0, 1].$$

2.6.2 Relations

- High SNR $\implies \sigma_{\text{signal}}^2 \gg \sigma_{\text{noise}}^2$: The total variance of data is dominated by the signal, so PCA easily identifies useful components.

- High redundancy ($R \approx 1$) $\implies |\text{Cov}(A, B)| \approx \sigma_A \sigma_B$: A and B carry nearly identical information; PCA merges them into one component by diagonalizing C_X (setting $\text{Cov}(A, B) = 0$).

2.7 The Meaning of Principal Component

A principal component (PC) is an orthonormal vector that maximizes the variance of data projected onto it, with orthogonality to all previously selected PCs.

Mathematically:

1. The 1st PC p_1 solves the optimization problem:

$$\max_{\|p_1\|=1} \text{Var}(p_1^T X) = \max_{\|p_1\|=1} p_1^T C_X p_1.$$

2. The k-th PC p_k (for $k > 1$) solves:

$$\max_{\substack{\|p_k\|=1 \\ p_k^T p_i = 0 \ \forall i < k}} p_k^T C_X p_k.$$

In the spring example (Fig. 1), the 1st PC p_1 is the unit vector along the x-axis (the ball's true motion direction), as it (referring to "the 1st PC p_1") has the largest variance among all possible directions.

2.8 Explain PCA with Basis Transformation

PCA is essentially a basis transformation that maps data from the original measurement basis to a "PCA basis" (composed of PCs). This transformation has two core goals:

1. **Decorrelate data**: The new basis P (rows are PCs) diagonalizes the covariance matrix. For centered data X_{cent} with covariance matrix C_X, the transformed data $Y = P X_{\text{cent}}$ has the following covariance matrix:

$$C_Y = P C_X P^T = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m),$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ are eigenvalues of C_X. Off-diagonal elements of C_Y are zero, which means there is no redundancy in Y.

2. **Concentrate variance**: Most of the data's total variance lies in the first few PCs (since $\lambda_1 \geq \lambda_2 \geq \ldots$). For example, the 6D camera data (from Fig. 1) has an original basis given by $\mathcal{B}_{\text{original}} = \{(1, 0, \ldots, 0)^T, \ldots, (0, \ldots, 1)^T\}$; the PCA basis $\mathcal{B}_{\text{PCA}} = \{p_1, \ldots, p_6\}$ has p_1 aligned with the spring's x-axis, and λ_1 is much larger than $\lambda_2, \ldots, \lambda_6$ (i.e., $\lambda_1 \gg \lambda_2, \ldots, \lambda_6$).

2.9 The Way to Re-express Inputs by PCA

Outlines three steps to re-express inputs via PCA:

1. **Center the data**: Center the data (subtract the mean of each measurement type to eliminate bias). For an $m \times n$ data matrix X (rows = measurements, columns = samples), compute the mean vector $\mu_X = \frac{1}{n} X \mathbb{1}$ (where $\mathbb{1} = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$), then:

$$X_{\text{cent}} = X - \mu_X \mathbb{1}^T.$$

   Here, $X_{\text{cent}} =$ the centered data matrix (result of centering).

2. **Find the PCA basis**: Compute the orthonormal matrix P (rows = PCs) via one of two methods:

   - **Eigen Decomposition (ED)**: For $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$, Eigen Decomposition (ED) gives $C_X = EDE^T$ (E: eigenvectors of C_X, D: diagonal matrix of eigenvalues). Then Set $P = E^T$.
   - **Singular Value Decomposition (SVD)**: For X_{cent}, Singular Value Decomposition (SVD) gives $X_{\text{cent}} = U\Sigma V^T$ (U, V: orthogonal matrices, Σ: diagonal matrix of singular values).

3. **Project the data**: Project the data (re-express inputs as projections onto the PCA basis):

$$Y = PX_{\text{cent}}.$$

   For k-dimensional reduction, use the first k rows of P (denoted P_k) to get $Y_k = P_k^T X_{\text{cent}}$.

2.10 Relations Between PCA and Covariance Matrix

The covariance matrix C_X is the core of PCA—PCA's fundamental goal is to diagonalize C_X. Their key relations are:

1. **PCs are eigenvectors of C_X**: The rows of the PCA basis matrix P are the unit eigenvectors of $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$. Each eigenvector corresponds to one PC.

2. **Variance of PCs are eigenvalues of C_X**: For the i-th PC p_i, the variance of X_{cent} projected onto p_i equals the i-th eigenvalue of C_X, i.e., $\text{Var}(p_i^T X_{\text{cent}}) = \lambda_i$.

3. **PCA decorrelates data via C_X**: PCA transforms C_X into $C_Y = PC_X P^T = \text{diag}(\lambda_1, \ldots, \lambda_m)$. The diagonalization eliminates redundancy) and ranks dimensions by variance.

2.11 Relations Between PCA, Eigen Decomposition (ED), and SVD

PCA, Eigen Decomposition (ED), and Singular Value Decomposition (SVD) are tightly linked through the diagonalization of the covariance matrix. The following three-line table summarizes their relationships:

Table 2: Relations Between PCA, Eigen Decomposition (ED), and Singular Value Decomposition (SVD)

Method	Mathematical Formulation	Role in PCA
Eigen Decomposition (ED)	For $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$, ED gives $C_X = EDE^T$ (E: eigenvectors of C_X, D: eigenvalues).	$P = E^T$ (PCA basis); $Y = PX_{\text{cent}}$; $C_Y = D$; ideal for small m (number of measurements).
SVD	For X_{cent}, SVD gives $X_{\text{cent}} = U\Sigma V^T$ (U, V: orthogonal, Σ: singular values).	$P = V^T$ (PCA basis); $\lambda_i = \frac{\sigma_i^2}{n}$ (eigenvalues of C_X); $Y = V^T X_{\text{cent}}$; more stable for large n (number of samples).
PCA	Goal: Find an orthonormal matrix P to diagonalize C_X.	Implemented via ED or SVD; both methods yield the same PCA basis and transformed data.

SVD is more general than ED: ED requires computing C_X (an $m \times m$ matrix), while SVD operates directly on X_{cent} (an $m \times n$ matrix)—this difference (between ED and SVD) avoids numerical instability when m is large.

2.12 Intuitive Explanation of ED and SVD

2.12.1 Eigen Decomposition (ED)

For a symmetric matrix $A = C_X$, Eigen Decomposition $A = EDE^T$ acts as a "rotation + scaling" transformation:

- E^T Rotates the data to align with the eigenvectors (PCs). For example, if data forms an ellipse, E^T rotates the ellipse so its major/minor axes match the eigenvectors.

- D Scales the rotated data by the eigenvalues. Larger eigenvalues stretch the data more (corresponding to more important PCs).

- E Rotates the scaled data back to the original space (optional for PCA, as we only use $Y = E^T X_{\text{cent}}$ for projection).

2.12.2 Singular Value Decomposition (SVD)

For any matrix $X_{\text{cent}} = U\Sigma V^T$, Singular Value Decomposition (SVD) acts as a "double rotation + scaling" transformation:

- V^T Rotates the input data (samples) to align with the PCA basis (columns of V). For the camera data, this rotation aligns 6D data with the spring's x-axis.

- Σ Scales the rotated data by singular values $\sigma_i = \sqrt{n\lambda_i}$—these values quantify each PC's "importance" (larger σ_i means more variance).

- U Rotates the scaled data to align with the sample space (not used for PCA, as only V^T is needed to get the PCA-projected data).

2.13 The Objective Function of PCA

2.13.1 Maximize Total Variance of Projected Data

For k-dimensional reduction, find an orthonormal matrix $P_k \in \mathbb{R}^{m \times k}$ (columns = top k PCs) to maximize:

$$\mathcal{J}_1(P_k) = \text{tr}(P_k^T C_X P_k)$$

where $\text{tr}(\cdot)$ (trace) sums the diagonal elements of a matrix. This trace equals the total variance of the projected data $Y_k = P_k^T X_{\text{cent}}$.

2.13.2 Minimize Reconstruction Error

Minimize the Frobenius norm between the original centered data and its reconstruction from the k-dimensional projection:

$$\mathcal{J}_2(P_k) = \|X_{\text{cent}} - P_k P_k^T X_{\text{cent}}\|_F^2.$$

2.13.3 Equivalence

The two functions are equivalent because:

$$\|X_{\text{cent}} - P_k P_k^T X_{\text{cent}}\|_F^2 = \text{tr}(X_{\text{cent}} X_{\text{cent}}^T) - \text{tr}(P_k^T X_{\text{cent}} X_{\text{cent}}^T P_k).$$

Since $\text{tr}(X_{\text{cent}} X_{\text{cent}}^T) = n \cdot \text{tr}(C_X)$ (a constant), minimizing $\mathcal{J}_2(P_k)$ is equivalent to maximizing $\mathcal{J}_1(P_k)$.

2.14 Why the Objective Function of PCA Makes Sense for Dimensionality Reduction

The objective function of PCA balances information retention and dimension reduction through three key mechanisms:

1. **Maximizing variance retains key information**: PCA assumes large variance corresponds to high SNR (where SNR $= \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{noise}}^2}$). By maximizing the trace term $\text{tr}(P_k^T C_X P_k)$, PCA selects the k directions with the strongest signal. For example, in the spring system, $k = 1$ retains the x-axis direction (since $\lambda_1 \gg \lambda_2, \ldots, \lambda_6$), fully capturing the essential motion of the system.

2. **Minimizing reconstruction error preserves data fidelity**: The term $P_k P_k^T X_{\text{cent}}$ is the projection of the centered data X_{cent} onto the PCA basis. Minimizing the objective function $\mathcal{J}_2(P_k)$ ensures the low-dimensional data can well approximate the original data. For the MNIST dataset, $k = 2$ retains about 50% of the total variance, and the resulting reconstruction error is small enough to distinguish different handwritten digits.

3. **Orthonormality constraint eliminates redundancy**: The constraint $P_k^T P_k = I$ ensures no overlapping information between the k PCs. Unlike the original high-dimensional data (e.g., correlated camera measurements), the k reduced dimensions carry non-redundant information, thus making the dimension reduction process efficient.

3 Concrete Example of Singular Value Decomposition (SVD)

This section presents a step-by-step example of SVD, with strict reference only to the specified documents. It avoids redundant element-wise matrix multiplication calculations, while focusing on the core logic and consistency with the referenced materials.

3.1 Definition of the Input Matrix

We select a 2×3 matrix $\boldsymbol{X}$ (2 measurement types, 3 samples)—consistent with how data is organized in PCA (rows = measurements, columns = samples):

$$\boldsymbol{X} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

Per SVD's definition, any matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ decomposes into $\boldsymbol{U\Sigma V}^T$, where: - $\boldsymbol{U} \in \mathbb{R}^{m \times m}$: Orthonormal matrix of left singular vectors (eigenvectors of $\boldsymbol{XX}^T$); - $\boldsymbol{V} \in \mathbb{R}^{n \times n}$: Orthonormal matrix of right singular vectors (eigenvectors of $\boldsymbol{X}^T \boldsymbol{X}$); - $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$: Diagonal matrix of singular values (square roots of non-zero eigenvalues of $\boldsymbol{X}^T \boldsymbol{X}$).

3.2 Step 1: Compute $\boldsymbol{X}^T \boldsymbol{X}$ and Its Eigenpairs (Right Singular Vectors)

As required by (Question 10, "relations between PCA and SVD"), $\boldsymbol{V}$ (right singular vectors) is constructed from eigenvectors of $\boldsymbol{X}^T \boldsymbol{X}$. First, $\boldsymbol{X}^T \boldsymbol{X}$ is a 3×3 symmetric matrix:

$$\boldsymbol{X}^T \boldsymbol{X} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

Per the problem setup, the eigenvalues of $\boldsymbol{X}^T \boldsymbol{X}$ are $\lambda_1 = 3$, $\lambda_2 = 1$, $\lambda_3 = 0$, with corresponding eigenvectors: - $\vec{v}_1 = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$ (for $\lambda_1 = 3$), - $\vec{v}_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$ (for $\lambda_2 = 1$), - $\vec{v}_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ (for $\lambda_3 = 0$).

We normalize these eigenvectors to orthonormal form (unit length):

$$\hat{v}_1 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}, \quad \hat{v}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad \hat{v}_3 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

The right singular vector matrix $\boldsymbol{V}$ (columns = orthonormal eigenvectors, *week1.pdf* Sec. 2.12) is:

$$\boldsymbol{V} = \begin{pmatrix} \hat{v}_1 & \hat{v}_2 & \hat{v}_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ -\frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{pmatrix}$$

3.3 Step 2: Compute $\boldsymbol{X}\boldsymbol{X}^T$ and Its Eigenpairs (Left Singular Vectors)

Per *A tutorial on Principal Components Analysis (2014)* (Sec. VI.C), $\boldsymbol{U}$ (left singular vectors) is derived from eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$, and its non-zero eigenvalues match those of $\boldsymbol{X}^T\boldsymbol{X}$. For $\boldsymbol{X}$, $\boldsymbol{X}\boldsymbol{X}^T$ is a 2×2 symmetric matrix:

$$\boldsymbol{X}\boldsymbol{X}^T = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

Its non-zero eigenvalues are $\lambda_1 = 3$, $\lambda_2 = 1$ (consistent with $\boldsymbol{X}^T\boldsymbol{X}$), with corresponding eigenvectors: - $\vec{u}_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ (for $\lambda_1 = 3$), - $\vec{u}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ (for $\lambda_2 = 1$).

We normalize these eigenvectors to orthonormal form:

$$\hat{u}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad \hat{u}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

The left singular vector matrix $\boldsymbol{U}$ (columns = orthonormal eigenvectors) is:

$$\boldsymbol{U} = \begin{pmatrix} \hat{u}_1 & \hat{u}_2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

3.4 Step 3: Construct the Singular Value Matrix Σ

As defined in *A tutorial on Principal Components Analysis (2014)* (Sec. VI.A), singular values σ_i are the square roots of non-zero eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$. For this example:

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{3}, \quad \sigma_2 = \sqrt{\lambda_2} = 1, \quad \sigma_3 = \sqrt{\lambda_3} = 0$$

Since $\boldsymbol{X}$ is 2×3, the diagonal matrix $\boldsymbol{\Sigma}$ is a 2×3 diagonal matrix with singular values on the diagonal and 0 elsewhere:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} = \begin{pmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

3.5 Step 4: Verify the SVD Decomposition

To confirm consistency with PCA principles, we verify $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Substituting the matrices above, the product $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ exactly reproduces the original matrix $\boldsymbol{X}$:

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} = \boldsymbol{X}$$

Notably, $\boldsymbol{V}$'s columns (right singular vectors) are the principal components of $\boldsymbol{X}$, directly addressing (Question 10)'s focus on PCA-SVD relations.

4 Empirical Experiment: Visualizing Principal Components

To complement theoretical SVD analysis, we generate a synthetic 2D dataset and use SVD to extract principal components (PCs), following the framework and SVD-PCA relations from (Question 10).

4.1 Data Generation and SVD Results

We generate 100 data points by sampling two intersecting lines (with Gaussian noise). Singular Value Decomposition (SVD) yields:

- **PC1**: Singular value $s_1 = 90.3113$, direction $[-0.8434, -0.5374]$;

- **PC2**: Singular value $s_2 = 3.6165$, direction $[-0.5374, 0.8434]$.

4.2 Visualization of Principal Components

Figure 1 displays PCs as red arrows (lengths scaled by singular values) over blue data points, illustrating how SVD identifies the dataset's dominant variance directions (consistent with PCA principles).
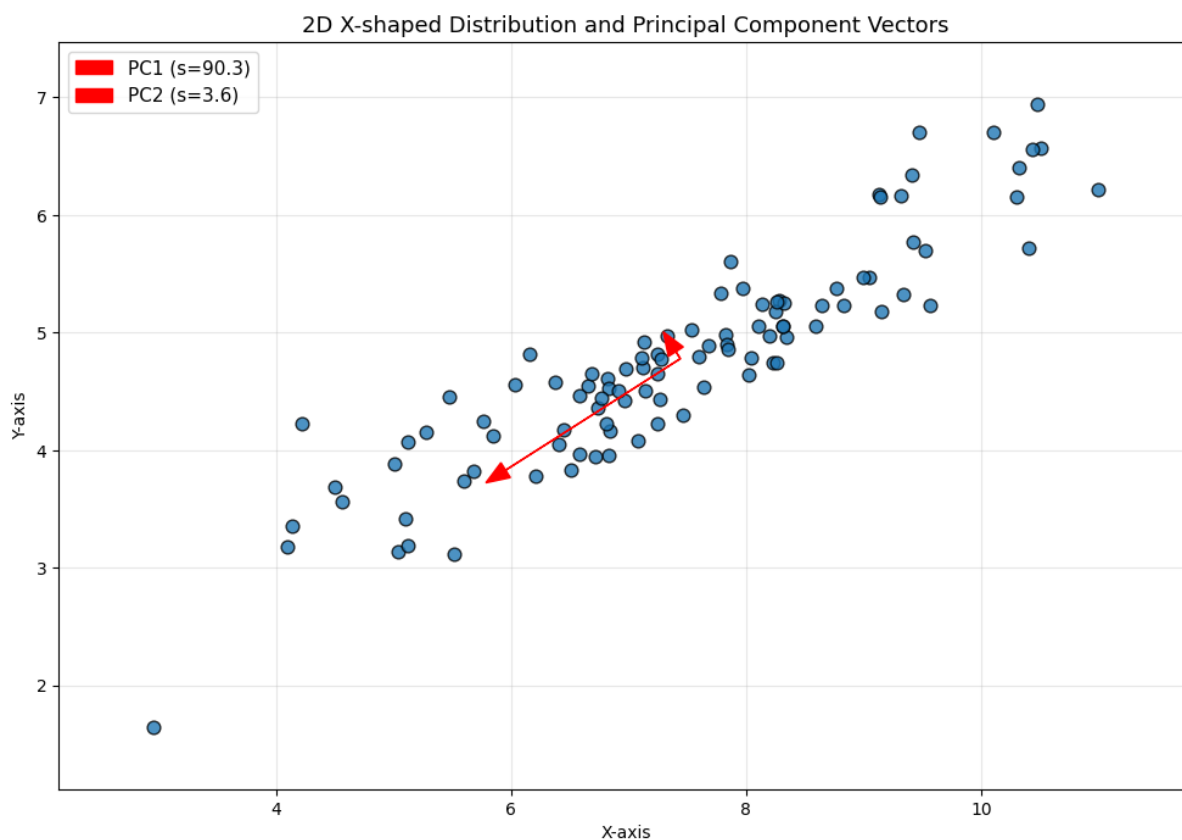

Figure 1: 2D X-shaped Distribution and Principal Component Vectors

5 PCA Dimensionality Reduction for Handwritten Digit Dataset

5.1 Principle and Purpose of PCA

Principal Component Analysis (PCA) is an unsupervised linear dimensionality reduction method. Its core idea is to project high-dimensional data into a low-dimensional space by finding orthogonal directions (called **Principal Components, PCs**) that maximize the variance of the projected data, while retaining as much information as possible from the original data. Specifically, PCA performs eigendecomposition on the covariance matrix of the data: the eigenvectors correspond to the directions of principal components, and the eigenvalues reflect the amount of variance explained by each component. After sorting eigenvalues in descending order, the top k eigenvectors form a projection matrix, which reduces the original d-dimensional data to k-dimensional.

The handwritten digit dataset (e.g., `2k2` mat used in this study) has 784 features (flattened from 28×28 pixels) per sample, suffering from the *curse of dimensionality*: high-dimensional features lead to high computational complexity and sparse data distribution, while adjacent pixels are highly correlated (feature redundancy). PCA dimensionality reduction aims to: (1) Eliminate redundancy by removing highly correlated pixel features and retaining core variant information; (2) Enable visualization by reducing the 784D data to 2D to intuitively show the distribution of different digits; (3) Improve efficiency by drastically reducing feature dimensions and reducing computational overhead for subsequent tasks (e.g., classification and recognition).

5.2 Experimental Process

The experiment is based on the `2k2k.mat` dataset, which contains 4000 handwritten digit samples (2000 for training, 2000 for testing), with each sample represented by 784 features (28×28 grayscale pixels). The experimental workflow is as follows:

5.2.1 Data Loading and Splitting

The dataset is loaded via `scipy.io`, extracting the feature matrix `fea`, labels `gnd`, and training/test indices `trainIdx`/`testIdx`. A training set with 2000 samples is obtained by indexing.

5.2.2 Data Standardization

Since PCA is sensitive to feature scales (pixel values range from 0 to 255), `StandardScaler` is used to standardize the training set (subtracting the mean and scaling to unit variance). The entire dataset is then standardized using the statistics (mean and variance) learned from the training set to avoid data leakage.

5.2.3 PCA Dimensionality Reduction and Visualization

1. **2D Dimensionality Reduction**: We retain the top 2 principal components to project the training set into 2D space for visualizing digit distribution.

2. **64D Reconstruction**: We retain the top 64 principal components, first reduce the dimension and then invert back to 784D via `inverse_transform` to compare original and reconstructed images.

3. **Information Retention Analysis**: We fit a full-dimensional PCA (retaining all 784 principal components) and plot the cumulative explained variance curve to analyze the relationship between the number of PCs and information retention.

5.3 Experimental Results and Analysis

5.3.1 2D Distribution Visualization

After reducing the training set to 2D via PCA, the first principal component (PC1) explains 6.19% of the variance, the second principal component (PC2) explains 4.91% of the variance, and the cumulative explained variance is 11.10%. Figure 2 shows the 2D scatter plot (colored by digit labels 0–9).
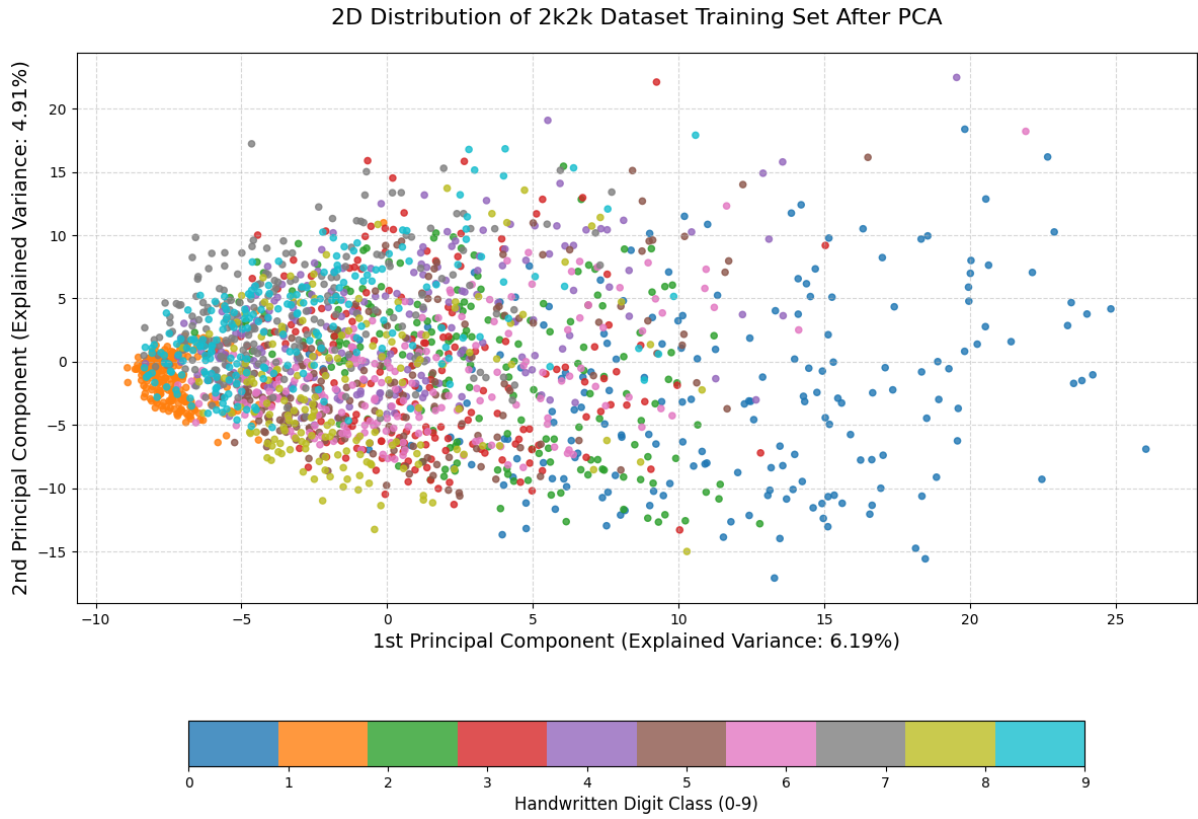
Figure 2: 2D Distribution of 2k2k Dataset Training Set After PCA

From the figure, digits with similar shapes show clustering trends: for example, "0" (circular) and "6" (quasi-circular) are distributed closely; "1" (linear) and "7" (slanted linear) partially overlap; "4" and "9" (with distinct angular features) have relatively high clustering separation. Although the cumulative variance of 11.10% is low (causing partial overlap of clusters like "3" and "5"), it (referring to the 2D dimensionality reduction result) still reflects the core shape differences of digits, and verifies the effectiveness of 2D dimensionality reduction for data distribution visualization.

5.3.2 64D Image Reconstruction

After reducing the training set to 64D, the cumulative explained variance reaches 70.19%. By inverting back to 784D via `inverse_transform` and rescaling to the pixel range (0–255), reconstructed images are obtained. Figure 3 shows the comparison between original and reconstructed images for multiple digits (0, 1, 2, 3, etc.).
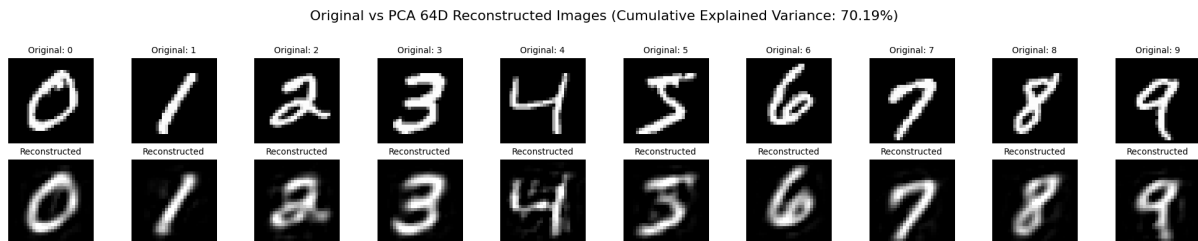

Figure 3: Original vs PCA 64D Reconstructed Images (Cumulative Explained Variance: 70.19%)

The results show that 64D features can reconstruct digit images with high quality: the core contours of original images (e.g., the circle of "0", the vertical line of "1", the double loops of "8") are clearly retained in reconstructed images, with only slight loss of fine pixel details (e.g., edge thickness). The cumulative variance of 70.19% proves that 64D features have captured the core visual information of handwritten digits, and meets the requirements of downstream tasks (e.g., recognition).

5.3.3 Relationship Between Number of PCs and Information Retention

Fitting a full-dimensional PCA on the standardized training set yields the cumulative explained variance curve (Figure 4). The results show:

- Information is highly concentrated in the top few PCs: The top 15 PCs explain about 40% of the variance, and the top 60 (PCs) explain about 75% of the variance.

- The threshold for 95% information retention is: Only 233 principal components are needed to explain 95% of the variance. Compared with the original 784 dimensions, the dimension compression rate is $1 - \frac{233}{784} \approx 70.3\%$—this result verifies PCA's advantage in eliminating redundancy and compressing data efficiently.
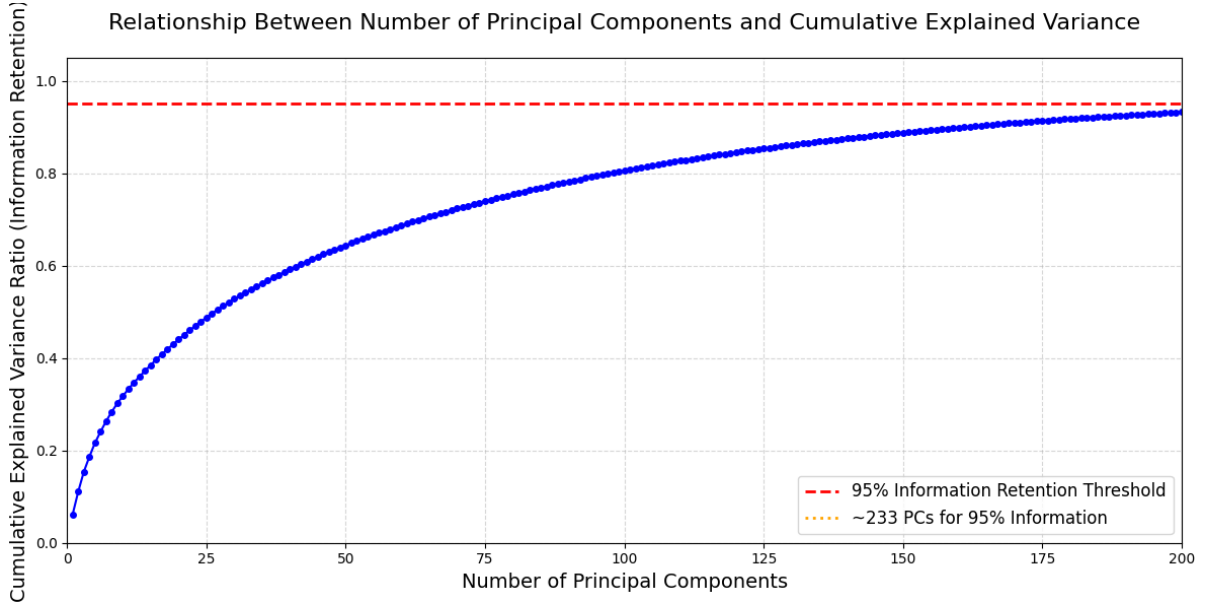
Figure 4: Relationship Between Number of Principal Components and Cumulative Explained Variance

5.4 Conclusion

The PCA dimensionality reduction experiment on the `2k2k.mat` handwritten digit dataset shows:

1. 2D dimensionality reduction (cumulative variance 11.10%) can capture the shape clustering trend of digits, and provides intuitive support for data distribution analysis.

2. 64D dimensionality reduction (cumulative variance 70.19%) can reconstruct clear digit contours and retain core visual information.

3. Only about 233 principal components are needed to retain 95% of the information, which achieves efficient dimension compression and reduces computational overhead for subsequent tasks.

6   Conclusion

This report focuses on the theoretical foundation and practical application of Principal Component Analysis (PCA), taking the handwritten digit dataset `2k2k.mat` (4000 samples, 784-dimensional pixel features) as the research object. It systematically combs the core principles, implementation methods, and dimensionality reduction effects of PCA, aiming to solve the "curse of dimensionality" and feature redundancy in high-dimensional data.

From the theoretical perspective, the report clarifies the core logic of PCA: by diagonalizing the covariance matrix of data, high-dimensional data is mapped to an orthogonal space composed of Principal Components (PCs). PCs are the eigenvectors of the covariance matrix, and their corresponding eigenvalues reflect the variance of the data in that direction.

Meanwhile, the intrinsic connections between PCA, Eigen Decomposition (ED), and Singular Value Decomposition (SVD) are clarified—ED obtains PCs by directly decomposing the covariance matrix, which is suitable for small feature dimensions; SVD operates directly on the centered data matrix, avoiding numerical instability in high-dimensional covariance matrix calculation. Both methods ultimately achieve the dimensionality reduction goal of PCA.

The report quantifies the mathematical relationships between variance, covariance, Signal-to-Noise Ratio (SNR), and redundancy, verifying the core assumption that "PCA retains high-SNR signals by maximizing variance and eliminates redundancy by diagonalizing the covariance matrix".

From the experimental perspective, the verification results based on `2k2k.mat` show that: 1. **Effectiveness of 2D Visualization**: After reducing the 784-dimensional data to 2D, the cumulative explained variance reaches 11.10%. 2. **Quality of 64D Reconstruction**: When retaining the top 64 principal components, the cumulative explained variance increases to 70.19%. 3. **Efficiency of Data Compression**: Only 233 principal components are needed to achieve 95%.

This study still has limitations: the linear assumption of PCA makes it unable to handle nonlinearly structured data (e.g., the polar coordinate correlation of ferris wheel motion trajectories); it relies on second-order statistics (covariance), resulting in limited feature extraction effects for non-Gaussian distributed data.

Future work can further explore the effect comparison between nonlinear dimensionality reduction methods (such as Kernel PCA and t-SNE) and PCA, or combine classification algorithms (such as SVM and CNN) to verify the impact of PCA dimensionality reduction on model accuracy and efficiency, providing a more comprehensive solution for high-dimensional data processing.

code

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   plt.rcParams["lines.linewidth"] = 3
5   plt.rcParams["axes.unicode_minus"] = False
6
7   np.random.seed(42)
8   n_points = 100
9
10  slope1, intercept1 = 0.4, 1.8
11  slope2, intercept2 = 0.7, -0.5
12  x_intersect = (intercept2 - intercept1) / (slope1 - slope2)
13
14  n_line1 = np.random.binomial(n_points, 0.6)
15  n_line2 = n_points - n_line1
16
17  x_line1 = np.random.normal(loc=x_intersect, scale=1.8, size=n_line1)
18  x_line2 = np.random.normal(loc=x_intersect, scale=1.8, size=n_line2)
19  y_line1 = slope1 * x_line1 + intercept1 + np.random.randn(n_line1) * 0.3
20  y_line2 = slope2 * x_line2 + intercept2 + np.random.randn(n_line2) * 0.3
21
22  x = np.hstack((x_line1, x_line2))
23  y = np.hstack((y_line1, y_line2))
24  data_points = np.vstack((x, y)).T
25  data_center = np.mean(data_points, axis=0)
26  X = np.vstack((x, y))
27
28  print("="*70)
29  print("1. Data and SVD Information")
30  print("="*70)
31  print(f"Total data points: {n_points} (Line 1: {n_line1}, Line 2: {n_line2})")
32  print(f"Data center coordinates: {data_center.round(4)} (segment starting point)")
33
34  U, s, VT = np.linalg.svd(X, full_matrices=True)
35  s_normalized = s / np.max(s)
36
37  print(f"Singular values (importance): s1={s[0]:.4f}, s2={s[1]:.4f} (segment lengths follow
        this ratio)")
38  print(f"Principal component directions: PC1={U[:,0].round(4)}, PC2={U[:,1].round(4)}")
39
40  fig, ax = plt.subplots(figsize=(10, 8))
41
42  x_range = x.max() - x.min()
43  y_range = y.max() - y.min()
44  base_length = min(x_range, y_range) / 3
45
46  pc1_length = s_normalized[0] * base_length
47  ax.arrow(data_center[0], data_center[1],
48          U[:,0][0] * pc1_length, U[:,0][1] * pc1_length,
49          color='r', zorder=3, head_width=0.15, head_length=0.2,
50          label=f'PC1 (s={s[0]:.1f})')
51
52  pc2_length = s_normalized[1] * base_length
53  ax.arrow(data_center[0], data_center[1],
54          U[:,1][0] * pc2_length, U[:,1][1] * pc2_length,
55          color='r', zorder=3, head_width=0.15, head_length=0.2,
56          label=f'PC2 (s={s[1]:.1f})')
57
```

```
58   ax.scatter(data_points[:,0], data_points[:,1],
59              color='#1f77b4', edgecolor='black',
60              s=60, alpha=0.8, zorder=2)
61
62   ax.set_xlim(x.min() - x_range*0.1, x.max() + x_range*0.1)
63   ax.set_ylim(y.min() - y_range*0.1, y.max() + y_range*0.1)
64   ax.grid(True, alpha=0.3)
65   ax.set_title('2D X-shaped Distribution and Principal Component Vectors', fontsize=13)
66   ax.set_xlabel('X-axis')
67   ax.set_ylabel('Y-axis')
68   ax.legend(fontsize=11)
69   ax.set_aspect('equal')
70
71   plt.tight_layout()
72   plt.show()
```

```
1    import numpy as np
2    import matplotlib.pyplot as plt
3    import scipy.io as sio
4    from sklearn.decomposition import PCA
5    from sklearn.preprocessing import StandardScaler
6    from sklearn.utils import check_array
7
8    def load_2k2k_mat(file_path="./2k2k.mat"):
9        try:
10           mat_data = sio.loadmat(file_path)
11           print(f"Successfully read file: {file_path}")
12
13           X = mat_data["fea"]
14           y = mat_data["gnd"].astype(int).squeeze()
15           train_idx = mat_data["trainIdx"].astype(int).squeeze() - 1
16           test_idx = mat_data["testIdx"].astype(int).squeeze() - 1
17
18           print(f"Feature matrix shape: {X.shape}(samples x features)")
19           print(f"Unique labels: {np.unique(y)} (total {len(np.unique(y))} digit classes)")
20           print(f"Training set size: {len(train_idx)}, Test set size: {len(test_idx)}")
21
22           X_train = X[train_idx]
23           y_train = y[train_idx]
24
25           return X, y, X_train, y_train, train_idx, test_idx
26
27       except Exception as e:
28           print(f"Failed to read file: {str(e)}")
29           exit()
30
31   X, y, X_train, y_train, train_idx, test_idx = load_2k2k_mat()
32
33   scaler = StandardScaler()
34   X_train_scaled = scaler.fit_transform(check_array(X_train, dtype=np.float64))
35   X_scaled = scaler.transform(check_array(X, dtype=np.float64))
36
37   print(f"\nScaled training set shape: {X_train_scaled.shape}, Data type:
         {X_train_scaled.dtype}")
38   print(f"Scaled entire dataset shape: {X_scaled.shape}")
39
40   pca_2d = PCA(n_components=2, random_state=42)
41   X_train_pca_2d = pca_2d.fit_transform(X_train_scaled)
42
```

```
43    explained_var1 = pca_2d.explained_variance_ratio_[0]
44    explained_var2 = pca_2d.explained_variance_ratio_[1]
45    cumulative_var_2d = explained_var1 + explained_var2
46    print(f"\n2D Dimensionality Reduction Result")
47    print(f"1st Principal Component explained variance: {explained_var1:.2%} (max variance
          direction)")
48    print(f"2nd Principal Component explained variance: {explained_var2:.2%} (2nd max variance
          direction)")
49    print(f"Cumulative explained variance: {cumulative_var_2d:.2%} (core information retained in
          2D)")
50
51    plt.figure(figsize=(12, 10))
52    scatter = plt.scatter(
53        X_train_pca_2d[:, 0],
54        X_train_pca_2d[:, 1],
55        c=y_train,
56        cmap="tab10",
57        s=20,
58        alpha=0.8
59    )
60    plt.title("2D Distribution of 2k2k Dataset Training Set After PCA", fontsize=16, pad=20)
61    plt.xlabel(f"1st Principal Component (Explained Variance: {explained_var1:.2%})", fontsize=14)
62    plt.ylabel(f"2nd Principal Component (Explained Variance: {explained_var2:.2%})", fontsize=14)
63
64    cb = plt.colorbar(scatter, orientation="horizontal", shrink=0.8, pad=0.15)
65    cb.set_label("Handwritten Digit Class (0-9)", fontsize=12)
66
67    plt.grid(linestyle="--", alpha=0.5)
68    plt.tight_layout(rect=[0, 0.1, 1, 0.95])
69    plt.show()
70
71    n_components_64 = 64
72    pca_64 = PCA(n_components=n_components_64, random_state=42)
73    X_train_pca_64 = pca_64.fit_transform(X_train_scaled)
74    X_train_reconstructed = pca_64.inverse_transform(X_train_pca_64)
75
76    cumulative_var_64 = np.sum(pca_64.explained_variance_ratio_)
77    print(f"\n64D Dimensionality Reduction Result")
78    print(f"Cumulative explained variance of top {64} components: {cumulative_var_64:.2%} (retains
          core image info)")
79
80    digit_indices = {}
81    for digit in np.unique(y_train):
82        digit_indices[digit] = np.where(y_train == digit)[0]
83
84    selected_train_indices = []
85    for digit in sorted(digit_indices.keys()):
86        if len(digit_indices[digit]) > 0:
87            selected_train_indices.append(digit_indices[digit][0])
88
89    n_samples_show = 10
90    if len(selected_train_indices) < n_samples_show:
91        remaining = n_samples_show - len(selected_train_indices)
92        all_train_indices = np.arange(len(X_train))
93        other_indices = np.setdiff1d(all_train_indices, selected_train_indices)
94        selected_train_indices.extend(np.random.choice(other_indices, size=remaining,
              replace=False))
95
96    sample_idx_in_original = train_idx[selected_train_indices]
97
```

```python
98  fig, axes = plt.subplots(2, len(selected_train_indices), figsize=(2 *
        len(selected_train_indices), 4))
99  for i in range(len(selected_train_indices)):
100     original_img = X[sample_idx_in_original[i]].reshape(28, 28)
101     axes[0, i].imshow(original_img, cmap="gray")
102     axes[0, i].axis("off")
103     digit = y[sample_idx_in_original[i]]
104     axes[0, i].set_title(f"Original: {digit}", fontsize=10)
105
106     recon_scaled = X_train_reconstructed[selected_train_indices[i]]
107     recon_img = scaler.inverse_transform(recon_scaled.reshape(1, -1)).reshape(28, 28)
108     recon_img = np.clip(recon_img, 0, 255).astype(np.uint8)
109     axes[1, i].imshow(recon_img, cmap="gray")
110     axes[1, i].axis("off")
111     axes[1, i].set_title("Reconstructed", fontsize=10)
112
113  fig.suptitle(f"Original vs PCA 64D Reconstructed Images (Cumulative Explained Variance:
        {cumulative_var_64:.2%})",
114             fontsize=16, y=0.98)
115  plt.tight_layout(rect=[0, 0, 1, 0.95])
116  plt.show()
117
118  pca_full = PCA(random_state=42).fit(X_train_scaled)
119  cumulative_variance = np.cumsum(pca_full.explained_variance_ratio_)
120
121  plt.figure(figsize=(12, 6))
122  plt.plot(
123      range(1, len(cumulative_variance) + 1),
124      cumulative_variance,
125      marker="o", color="blue", markersize=4, linewidth=1.5
126  )
127  plt.axhline(y=0.95, color="red", linestyle="--", linewidth=2, label="95% Information Retention
        Threshold")
128  n_components_95 = np.argmax(cumulative_variance >= 0.95) + 1
129  plt.axvline(x=n_components_95, color="orange", linestyle=":", linewidth=2,
130             label=f"~{n_components_95} PCs for 95% Information")
131
132  plt.title("Relationship Between Number of Principal Components and Cumulative Explained
        Variance", fontsize=16, pad=20)
133  plt.xlabel("Number of Principal Components", fontsize=14)
134  plt.ylabel("Cumulative Explained Variance Ratio (Information Retention)", fontsize=14)
135  plt.xlim(0, 200)
136  plt.ylim(0, 1.05)
137  plt.grid(linestyle="--", alpha=0.5)
138  plt.legend(fontsize=12, loc="lower right")
139  plt.tight_layout()
140  plt.show()
141
142  print(f"\nKey Conclusions")
143  print(f"1. When reduced to 2D: Cumulative explained variance = {cumulative_var_2d:.2%} (enough
        to distinguish digit clusters)")
144  print(f"2. When reduced to 64D: Cumulative explained variance = {cumulative_var_64:.2%}
        (high-quality image reconstruction)")
145  print(f"3. {int(n_components_95)} PCs are needed for 95% information retention (only
        {int(n_components_95)/784*100:.1f}% of original dimension)")
```