Report on Principal Component Analysis (PCA)

Xiong Rui

October 24, 2025

1 Mathematical Symbols Summary

Table 1: 1. Summary of Scalar Variables

No.	Notation	Definition
1	A	Zero-mean variable set: $A = \{a_1, a_2, ..., a_n\}$, used to calculate covariance with variable set B.
2	B	Zero-mean variable set: $B = \{b_1, b_2, ..., b_n\}$, used to calculate covariance with variable set A.
3	λ_i	Eigenvalue of covariance matrix C_X, equal to the variance of centered data X_{cent} projected onto the i-th principal component p_i. Sorted in non-increasing order: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$.
4	m	Number of measurement types (corresponds to rows of data matrix X, where X is an $m \times n$ matrix).
5	n	Number of samples (corresponds to columns of data matrix X, where X is an $m \times n$ matrix).
6	r	Rank of centered data matrix X_{cent} (equal to the number of non-zero singular values in singular value matrix Σ).
7	SNR	Signal-to-Noise Ratio: Defined as SNR $= \frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{noise}}}$. A value of SNR $\gg 1$ indicates a strong signal relative to measurement noise.
8	σ^2_A	Variance of zero-mean variable A: Calculated as $\sigma^2_A = \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \text{Cov}(A, A)$.
9	σ^2_{noise}	Noise variance: Originates from measurement errors (e.g., camera noise in the spring toy example), serving as the denominator in the SNR formula.
10	σ^2_{signal}	Signal variance: Originates from the true structure of the data (e.g., spring motion), serving as the numerator in the SNR formula.
11	σ_i	Singular value of centered data matrix X_{cent}: Diagonal entry of singular value matrix Σ (from SVD), satisfying $\sigma_i = \sqrt{n\lambda_i}$.

Table 2: 2. Summary of Vectors

No.	Notation	Definition
1	μ_X	Mean vector of data matrix X: An m-dimensional column vector, calculated as $\mu_X = \frac{1}{n}X\mathbf{1}_n$ (where $\mathbf{1}_n$ is an n-dimensional column vector with all elements 1). Each row corresponds to the mean of one measurement type in X.
2	p_i	i-th principal component (PC): The i-th row of orthonormal basis matrix P ($p_i \in \mathbb{R}^m$), which is an eigenvector of covariance matrix C_X and satisfies orthogonality: $p_i^T p_j = \delta_{ij}$.
3	$\vec{X}$	Sample column vector of data matrix X: Represents one experimental sample (e.g., 6D position data from 3 cameras in the spring toy example).

Table 3: 3. Summary of Matrices

No.	Notation	Definition
1	C_X	Covariance matrix of data matrix X: An $m \times m$ symmetric matrix, calculated for centered data X_{cent} as $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$.
2	D	Eigenvalue matrix of covariance matrix C_X: A diagonal matrix with eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$ (of C_X) as its diagonal entries.
3	E	Eigenvector matrix of covariance matrix C_X: An $m \times m$ orthogonal matrix, satisfying $C_X = EDE^T$ (columns of E are eigenvectors of C_X).
4	I	Identity matrix: Dimension depends on context. All diagonal entries are 1, and off-diagonal entries are 0.
5	P	Orthonormal basis matrix for PCA: An $m \times m$ matrix whose rows are principal components (PCs), satisfying $PP^T = I$.
6	Σ	Singular value matrix in SVD: An $m \times n$ diagonal matrix with entries $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$ (singular values of X_{cent}, where r is the rank of X_{cent}), used in the SVD decomposition $X_{\text{cent}} = U\Sigma V^T$.
7	U	Orthogonal matrix in SVD: An $m \times m$ matrix satisfying $UU^T = I$, part of the SVD decomposition $X_{\text{cent}} = U\Sigma V^T$.
8	V	Orthogonal matrix in SVD: An $n \times n$ matrix satisfying $VV^T = I$, part of the SVD decomposition $X_{\text{cent}} = U\Sigma V^T$. Columns of V are the principal components of X.
9	X	Data matrix: An $m \times n$ matrix where m = number of measurement types and n = number of samples. Each column is a single sample $\vec{X}$.
10	X_{cent}	Centered data matrix: An $m \times n$ matrix calculated as $X_{\text{cent}} = X - \mu_X \cdot \mathbf{1}_n^T$ (where $\mathbf{1}_n^T$ is an n-dimensional row vector with all elements 1). Each row of X_{cent} has zero mean.
11	Y	PCA-transformed data matrix: An $m \times n$ matrix calculated as $Y = PX_{\text{cent}}$, whose rows are the projections of X_{cent} onto the principal components (PCs).
12	Y_k	k-dimensional reduced data matrix: A $k \times n$ matrix calculated as $Y_k = P_k^T X_{\text{cent}}$, where P_k is the matrix consisting of the first k columns of V (from SVD).
13	Y_{SVD}	SVD auxiliary matrix: An $n \times m$ matrix calculated as $Y_{\text{SVD}} = \frac{1}{\sqrt{n}} X_{\text{cent}}^T$, satisfying $Y_{\text{SVD}}^T Y_{\text{SVD}} = C_X$ (links SVD to the covariance matrix C_X).

Table 4: 4. Summary of Operators & Special Symbols

No.	Notation	Definition
1	$\text{Cov}(A, B)$	Covariance of variables A and B: Calculated as $\text{Cov}(A, B) = \frac{1}{n} \sum_{i=1}^{n} a_i b_i$, quantifying the linear correlation between A and B.
2	δ_{ij}	Kronecker delta: Defined as $\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$ otherwise. Describes the orthogonality of principal components (e.g., $p_i^T p_j = \delta_{ij}$).
3	$\|\cdot\|_F$	Frobenius norm of a matrix: Calculated as $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$, used to define the reconstruction error of PCA.
4	$\text{tr}(\cdot)$	Trace of a square matrix: The sum of the matrix's diagonal entries. For covariance matrix C_X, $\text{tr}(C_X) = \sum_{i=1}^{m} \lambda_i$ (representing the total variance of the data).

2 Answers to Q&A

2.1 The Essence of Matrix

From a linear algebra perspective, 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**: Reducing 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 linear basis transformation. For original data X (centered) and transformed data Y, the relation is $Y = PX$, where P is an orthonormal matrix (rows = principal components). Linear transformation simplifies the problem to "basis change" (a core concept in linear algebra), which has clear analytical solutions (e.g., eigen decomposition, SVD). Without linearity, PCA would need complex nonlinear tools, losing its simplicity and interpretability.
2. **Large variance implies important structure**: For the i-th principal component p_i, its associated eigenvalue $\lambda_i = \text{Var}(p_i^T X)$ (eigenvalue of covariance matrix C_X) reflects signal strength. Useful components satisfy:
$$\text{SNR}_i = \frac{\lambda_i - \sigma^2_{\text{noise}}}{\sigma^2_{\text{noise}}} \gg 1,$$
where σ^2_{noise} is noise variance. *Assumption reason*: Noise usually has small, random variance, while real signals (e.g., spring motion in the toy example) contribute large variance. This assumption lets PCA filter noise by focusing on high-variance directions.
3. **Orthonormality of PCs**: Principal components satisfy $p_i^T p_j = \delta_{ij}$ (Kronecker delta: 1 if $i = j$, 0 otherwise), meaning distinct PCs are orthogonal and have unit length. *Assumption reason*: Orthogonality eliminates redundancy between PCs (decorrelates data), making the covariance matrix of Y diagonal. It also ensures an efficient analytical solution (e.g., eigenvectors of C_X are orthogonal), avoiding overlapping information.

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, second-order statistics (covariance) fail to capture higher-order dependencies, leading to suboptimal dimensionality reduction.

2.4 Basis in Linear Algebra

To elaborate on the linear algebra concept of "basis", the following first explains its core intuitively, then defines it mathematically rigorously, and ends with a camera data example to deepen understanding.

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

Covariance not only quantifies linear correlation between variables but also serves as a core link between data redundancy, noise, and PCA's subsequent redundancy elimination and noise filtering.

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:

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

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. **Correlational Redundancy**: 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**: 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:

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

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

$$\mathrm{SNR} = \frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{noise}}},$$

where σ^2_{signal} is signal variance and σ^2_{noise} is noise variance.

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

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

2.6.2 Relations

- High SNR $\implies \sigma^2_{\text{signal}} \gg \sigma^2_{\text{noise}}$: The total variance of data is dominated by the signal, so PCA easily identifies useful components.
- High redundancy ($R \approx 1$) $\implies |\mathrm{Cov}(A, B)| \approx \sigma_A \sigma_B$: A and B carry nearly identical information. PCA merges them into one component by diagonalizing C_X (setting $\mathrm{Cov}(A, B) = 0$).

2.7 The Meaning of Principal Components

PCA chooses to maximize covariance because useful signals in data (such as the motion of a spring-attached ball along the x-axis) contribute structured large variance, while noise only adds small random variance. Maximizing covariance allows the selected direction to capture more of these useful signals and filter out noise. For example, the direction of the ball's motion has the largest covariance, which corresponds to the real dynamic information of the system.

PCA adopts orthogonal bases because non-orthogonal basis vectors have non-zero covariance, which leads to information redundancy in data (such as the highly correlated x-coordinates measured by two adjacent cameras). Orthogonal bases (satisfying $p_i^T p_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta) can make the covariance matrix of the transformed data diagonal, thereby completely eliminating redundancy. Meanwhile, orthogonality enables the basis vectors to be directly solved through eigen decomposition or singular value decomposition (SVD) of the covariance matrix, ensuring simple calculation and unique results.

A principal component (PC) is a set of orthogonal bases where each basis vector maximizes the covariance of the data projected onto that direction, and each newly added basis vector is orthogonal to all previously selected basis vectors.

Mathematically:

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

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

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 $Y = PX_{\text{cent}}$ has the following covariance matrix:
$$C_Y = PC_XP^T = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m),$$
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. For example, the 6D camera data 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 $\mu_X = \frac{1}{n}X\mathbf{1}_n$ (where $\mathbf{1}_n$ is an n-dimensional column vector with all elements 1), then:
$$X_{\text{cent}} = X - \mu_X\mathbf{1}_n^T.$$
Here, X_{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 $C_Y = PC_XP^T = \text{diag}(\lambda_1, \ldots, \lambda_m)$. 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 inherently connected through the core goal of PCA—diagonalizing the data's covariance matrix. Their relations are mainly reflected in mathematical form correspondence and consistent principle of diagonalization, as summarized in the table below:

Table 5: Relations Between PCA, Eigen Decomposition (ED), and Singular Value Decomposition (SVD)

Method	Mathematical Form Correspondence	Principle-Level Relation
ED	Targets the covariance matrix C_X (where $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$). Decomposes C_X into $C_X = EDE^T$ (E: orthogonal eigenvector matrix, D: diagonal eigenvalue matrix).	Serves as PCA's direct mathematical basis: PCA's orthonormal matrix $P = E^T$ (rows of P = columns of E), and D is the covariance matrix of PCA-transformed data Y (i.e., $C_Y = D$), directly realizing C_X diagonalization.
SVD	Targets the centered data matrix X_{cent}. Decomposes it into $X_{\text{cent}} = U\Sigma V^T$ (U, V: orthogonal matrices, Σ: diagonal singular value matrix).	Links to PCA indirectly via C_X: V (right singular vectors of X_{cent}) equals E (eigenvectors of C_X), eigenvalues satisfy $\lambda_i = \frac{\sigma_i^2}{n}$ (σ_i: singular values in Σ), and $P = V^T$ to achieve C_X diagonalization.
PCA	Core requirement: Find an orthonormal matrix P such that $PC_XP^T = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ (diagonal matrix of eigenvalues).	Unifies ED and SVD as implementation paths: Both methods provide the P needed for C_X diagonalization, with mathematically consistent results (same P and C_Y).

In essence, ED and SVD are two linear algebra tools that "realize PCA's goal through different decomposition objects": ED directly decomposes the covariance matrix C_X to get P, while SVD decomposes the centered data matrix X_{cent} and indirectly associates with C_X via $C_X = \frac{1}{n} X_{\text{cent}} X_{\text{cent}}^T$. This makes SVD more general—since it avoids direct computation of C_X, it is more suitable for high-dimensional data scenarios, as noted in the document's discussion of SVD's advantages over ED.

2.12 Intuitive Explanation of ED and SVD

We unpack the intuitive transformation meanings of ED and SVD, matching their application logic.

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 axes match the eigenvectors.
- D Scales the rotated data by the eigenvalues. Larger eigenvalues stretch the data more.
- 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

PCA's objective functions mainly include maximizing the total variance of projected data and minimizing reconstruction error, with the two being mathematically equivalent.

2.13.1 Maximize Total Variance of Projected Data

For k-dimensional reduction, the objective of PCA is to find an orthonormal matrix $P_k \in \mathbb{R}^{m\times k}$ (columns = top k principal components, PCs) that maximizes the total variance of the projected data. This objective is mathematically expressed as the following optimization problem:

$$\max_{P_k\in\mathbb{R}^{m\times k},\ P_k^T P_k=I_k} \mathcal{J}_1(P_k) = \mathrm{tr}(P_k^T C_X P_k),$$

where: The constraint $P_k^T P_k = I_k$ (I_k denotes the k-dimensional identity matrix) ensures the columns of P_k (i.e., the top k PCs) are orthonormal, consistent with PCA's assumption of orthogonal principal components. $\mathrm{tr}(\cdot)$ (trace) is the sum of the diagonal elements of a matrix. The trace term $\mathrm{tr}(P_k^T C_X P_k)$ 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:

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

Since $\mathrm{tr}(X_{\text{cent}} X_{\text{cent}}^T) = n \cdot \mathrm{tr}(C_X)$ (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 = $\frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{noise}}}$). By maximizing the trace term $\mathrm{tr}(P_k^T C_X P_k)$, 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 orthonormality constraint ($P_k^T P_k = I$, P_k = matrix of top k PCs) eliminates this redundancy by diagonalizing the covariance matrix C_X of centered data X_{cent}.

   When projecting X_{cent} to low dimension via $Y_k = P_k^T X_{\text{cent}}$, the orthonormal P_k makes the covariance matrix of Y_k ($C_{Y_k} = P_k^T C_X P_k$) diagonal. This sets all off-diagonal elements (covariances between PCs) to zero, removing overlapping information—thus eliminating redundancy.

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 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where: - $\boldsymbol{U} \in \mathbb{R}^{m \times m}$: Orthonormal matrix of left singular vectors (eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$). - $\boldsymbol{V} \in \mathbb{R}^{n \times n}$: 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 $\boldsymbol{\Sigma}$

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.

The schematic diagram of the above decomposition process is as follows:

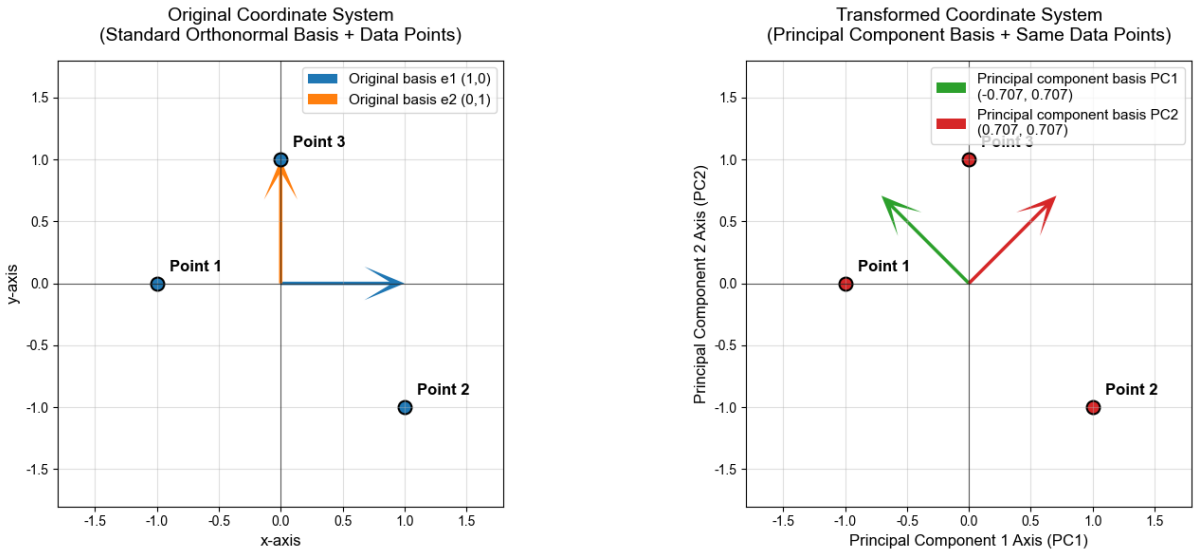

Figure 1: SVD Decomposition Verification

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 2 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).

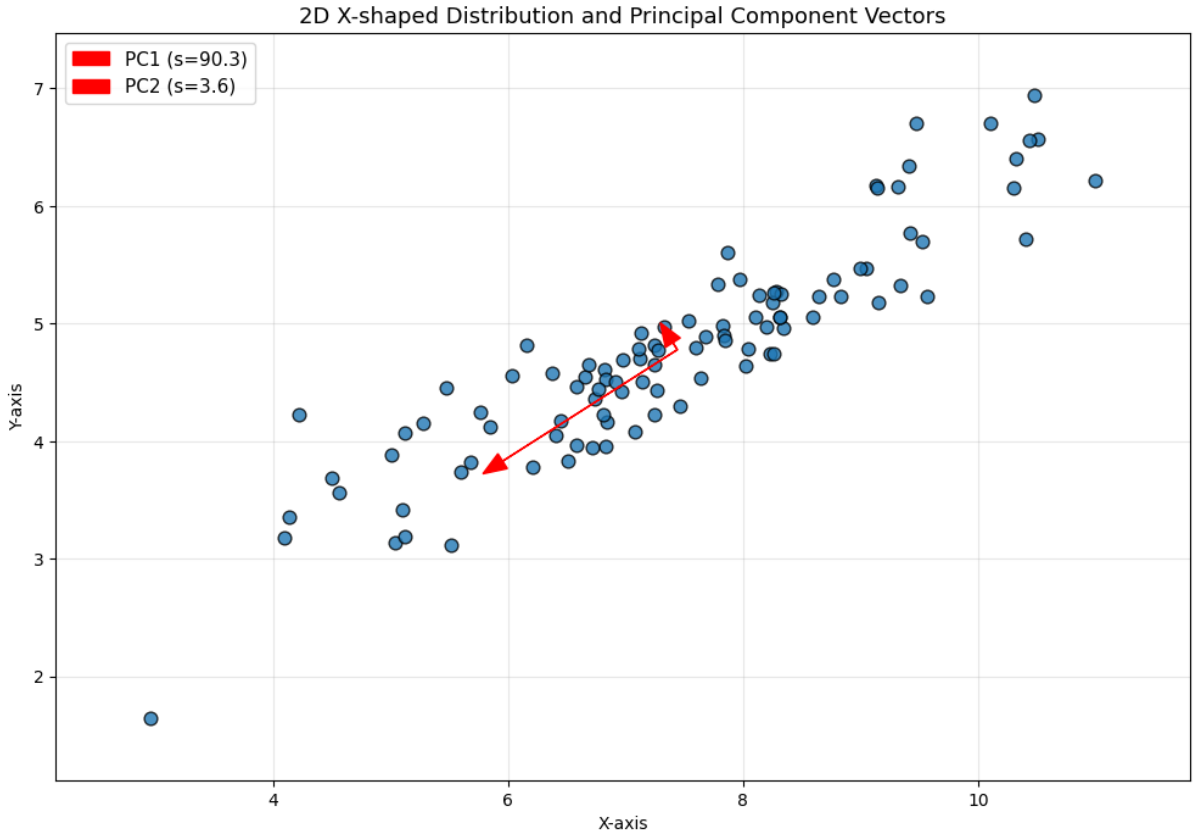

Figure 2: 2D X-shaped Distribution and Principal Component Vectors

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 their corresponding 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., `MNIST` 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 `MNIST` 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 3 shows the 2D scatter plot (colored by digit labels 0–9).

From the 2D distribution of the MNIST Dataset Training, key observations are as follows:

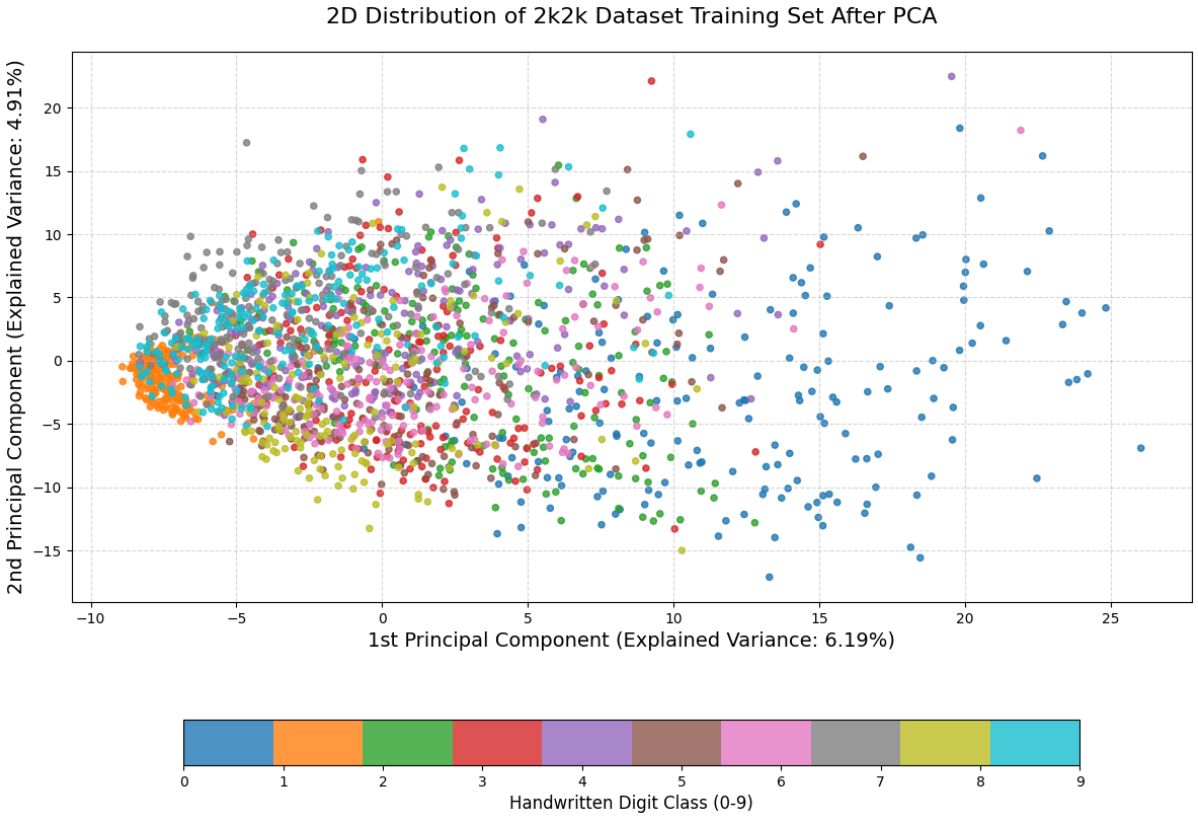

Figure 3: 2D Distribution of 2k2k Dataset Training Set After PCA

Different digits (0-9) are marked with distinct colors in the 2D visualization, and their distribution aligns closely with their intrinsic shape characteristics. Digits with similar morphological features show clear clustering trends: "0" (characterized by a closed circular contour) and "6" (with a quasi-circular upper part and a vertical tail) are closely distributed. "1" (dominated by a single vertical linear feature) and "7" (featuring a horizontal top plus a slanted linear segment) partially overlap. "8" (with two interconnected circular loops) and "9" (with a circular top and a downward tail) also exhibit proximity due to their shared circular structural elements. In contrast, "4" (with distinct right angles and a vertical line) and "9" maintain relatively clearer separation.All these patterns are consistent with the tutorial's assumption that directions with larger variance capture key data structures.

Some clusters (e.g., "3" and "5") show overlap, which can be attributed to the low cumulative variance explained by the top 2 principal components (11.10%).

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 4 shows the comparison between original and reconstructed images for multiple digits.

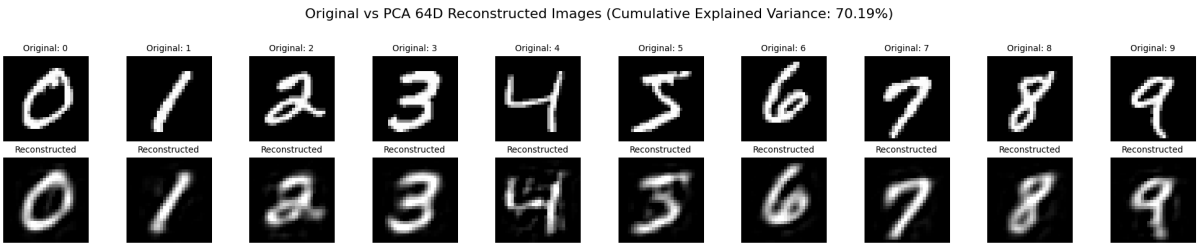

Figure 4: Original vs PCA 64D Reconstructed Images (Cumulative Explained Variance: 70.19%)

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 5). 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 $\mathbf{1 - \frac{233}{784} \approx 70.3\%}$—this result verifies PCA's advantage in eliminating redundancy and compressing data efficiently.

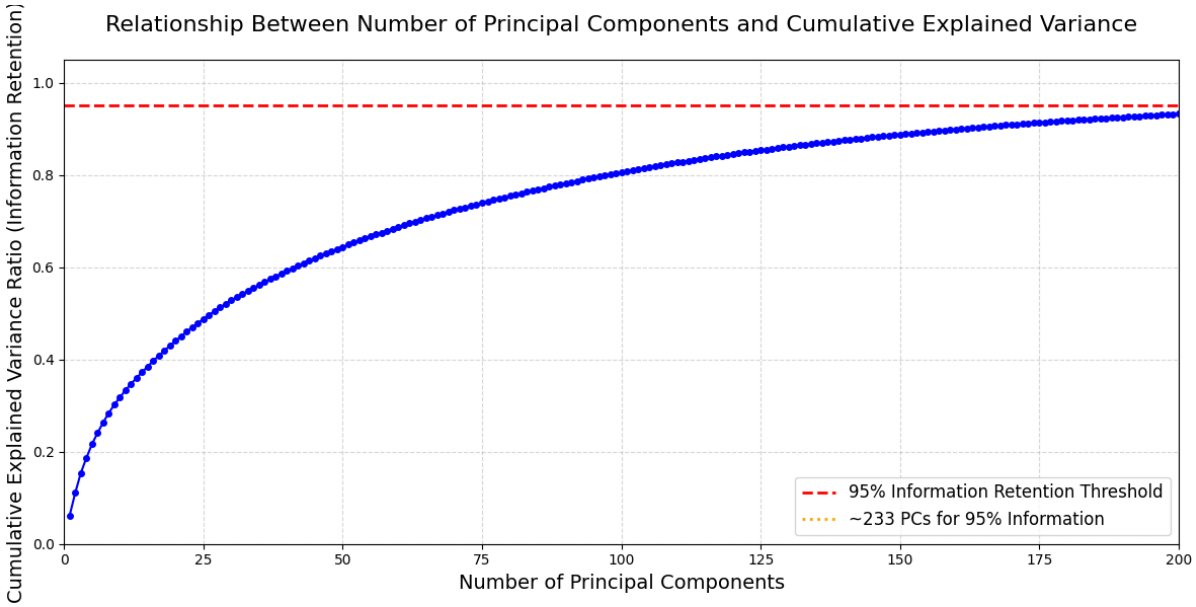

Figure 5: Relationship Between Number of Principal Components and Cumulative Explained Variance

5.4 Conclusion

The PCA dimensionality reduction experiment on the `MNIST` 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 `MNIST` (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 `MNIST` 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

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams["lines.linewidth"] = 3
plt.rcParams["axes.unicode_minus"] = False

np.random.seed(42)
n_points = 100

slope1, intercept1 = 0.4, 1.8
slope2, intercept2 = 0.7, -0.5
x_intersect = (intercept2 - intercept1) / (slope1 - slope2)

n_line1 = np.random.binomial(n_points, 0.6)
n_line2 = n_points - n_line1

x_line1 = np.random.normal(loc=x_intersect, scale=1.8, size=n_line1)
x_line2 = np.random.normal(loc=x_intersect, scale=1.8, size=n_line2)
y_line1 = slope1 * x_line1 + intercept1 + np.random.randn(n_line1) * 0.3
y_line2 = slope2 * x_line2 + intercept2 + np.random.randn(n_line2) * 0.3

x = np.hstack((x_line1, x_line2))
y = np.hstack((y_line1, y_line2))
data_points = np.vstack((x, y)).T
data_center = np.mean(data_points, axis=0)
X = np.vstack((x, y))

print("="*70)
print("1. Data and SVD Information")
print("="*70)
print(f"Total data points: {n_points} (Line 1: {n_line1}, Line 2: {n_line2})")
print(f"Data center coordinates: {data_center.round(4)} (segment starting point)")

U, s, VT = np.linalg.svd(X, full_matrices=True)
s_normalized = s / np.max(s)

print(f"Singular values (importance): s1={s[0]:.4f}, s2={s[1]:.4f} (segment lengths follow
    this ratio)")
print(f"Principal component directions: PC1={U[:,0].round(4)}, PC2={U[:,1].round(4)}")

fig, ax = plt.subplots(figsize=(10, 8))

x_range = x.max() - x.min()
y_range = y.max() - y.min()
base_length = min(x_range, y_range) / 3

pc1_length = s_normalized[0] * base_length
ax.arrow(data_center[0], data_center[1],
        U[:,0][0] * pc1_length, U[:,0][1] * pc1_length,
        color='r', zorder=3, head_width=0.15, head_length=0.2,
        label=f'PC1 (s={s[0]:.1f})')

pc2_length = s_normalized[1] * base_length
ax.arrow(data_center[0], data_center[1],
        U[:,1][0] * pc2_length, U[:,1][1] * pc2_length,
        color='r', zorder=3, head_width=0.15, head_length=0.2,
        label=f'PC2 (s={s[1]:.1f})')

```

```
ax.scatter(data_points[:,0], data_points[:,1],
           color='#1f77b4', edgecolor='black',
           s=60, alpha=0.8, zorder=2)

ax.set_xlim(x.min() - x_range*0.1, x.max() + x_range*0.1)
ax.set_ylim(y.min() - y_range*0.1, y.max() + y_range*0.1)
ax.grid(True, alpha=0.3)
ax.set_title('2D X-shaped Distribution and Principal Component Vectors', fontsize=13)
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.legend(fontsize=11)
ax.set_aspect('equal')

plt.tight_layout()
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.utils import check_array

def load_2k2k_mat(file_path="./2k2k.mat"):
    try:
        mat_data = sio.loadmat(file_path)
        print(f"Successfully read file: {file_path}")

        X = mat_data["fea"]
        y = mat_data["gnd"].astype(int).squeeze()
        train_idx = mat_data["trainIdx"].astype(int).squeeze() - 1
        test_idx = mat_data["testIdx"].astype(int).squeeze() - 1

        print(f"Feature matrix shape: {X.shape}(samples x features)")
        print(f"Unique labels: {np.unique(y)} (total {len(np.unique(y))} digit classes)")
        print(f"Training set size: {len(train_idx)}, Test set size: {len(test_idx)}")

        X_train = X[train_idx]
        y_train = y[train_idx]

        return X, y, X_train, y_train, train_idx, test_idx

    except Exception as e:
        print(f"Failed to read file: {str(e)}")
        exit()

X, y, X_train, y_train, train_idx, test_idx = load_2k2k_mat()

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(check_array(X_train, dtype=np.float64))
X_scaled = scaler.transform(check_array(X, dtype=np.float64))

print(f"\nScaled training set shape: {X_train_scaled.shape}, Data type:
    {X_train_scaled.dtype}")
print(f"Scaled entire dataset shape: {X_scaled.shape}")

pca_2d = PCA(n_components=2, random_state=42)
X_train_pca_2d = pca_2d.fit_transform(X_train_scaled)

```

```
explained_var1 = pca_2d.explained_variance_ratio_[0]
explained_var2 = pca_2d.explained_variance_ratio_[1]
cumulative_var_2d = explained_var1 + explained_var2
print(f"\n2D Dimensionality Reduction Result")
print(f"1st Principal Component explained variance: {explained_var1:.2%} (max variance direction)")
print(f"2nd Principal Component explained variance: {explained_var2:.2%} (2nd max variance direction)")
print(f"Cumulative explained variance: {cumulative_var_2d:.2%} (core information retained in 2D)")

plt.figure(figsize=(12, 10))
scatter = plt.scatter(
    X_train_pca_2d[:, 0],
    X_train_pca_2d[:, 1],
    c=y_train,
    cmap="tab10",
    s=20,
    alpha=0.8
)
plt.title("2D Distribution of 2k2k Dataset Training Set After PCA", fontsize=16, pad=20)
plt.xlabel(f"1st Principal Component (Explained Variance: {explained_var1:.2%})", fontsize=14)
plt.ylabel(f"2nd Principal Component (Explained Variance: {explained_var2:.2%})", fontsize=14)

cb = plt.colorbar(scatter, orientation="horizontal", shrink=0.8, pad=0.15)
cb.set_label("Handwritten Digit Class (0-9)", fontsize=12)

plt.grid(linestyle="--", alpha=0.5)
plt.tight_layout(rect=[0, 0.1, 1, 0.95])
plt.show()

n_components_64 = 64
pca_64 = PCA(n_components=n_components_64, random_state=42)
X_train_pca_64 = pca_64.fit_transform(X_train_scaled)
X_train_reconstructed = pca_64.inverse_transform(X_train_pca_64)

cumulative_var_64 = np.sum(pca_64.explained_variance_ratio_)
print(f"\n64D Dimensionality Reduction Result")
print(f"Cumulative explained variance of top {64} components: {cumulative_var_64:.2%} (retains core image info)")

digit_indices = {}
for digit in np.unique(y_train):
    digit_indices[digit] = np.where(y_train == digit)[0]

selected_train_indices = []
for digit in sorted(digit_indices.keys()):
    if len(digit_indices[digit]) > 0:
        selected_train_indices.append(digit_indices[digit][0])

n_samples_show = 10
if len(selected_train_indices) < n_samples_show:
    remaining = n_samples_show - len(selected_train_indices)
    all_train_indices = np.arange(len(X_train))
    other_indices = np.setdiff1d(all_train_indices, selected_train_indices)
    selected_train_indices.extend(np.random.choice(other_indices, size=remaining, replace=False))

sample_idx_in_original = train_idx[selected_train_indices]

```

```
fig, axes = plt.subplots(2, len(selected_train_indices), figsize=(2 * len(selected_train_indices), 4))
for i in range(len(selected_train_indices)):
    original_img = X[sample_idx_in_original[i]].reshape(28, 28)
    axes[0, i].imshow(original_img, cmap="gray")
    axes[0, i].axis("off")
    digit = y[sample_idx_in_original[i]]
    axes[0, i].set_title(f"Original: {digit}", fontsize=10)

    recon_scaled = X_train_reconstructed[selected_train_indices[i]]
    recon_img = scaler.inverse_transform(recon_scaled.reshape(1, -1)).reshape(28, 28)
    recon_img = np.clip(recon_img, 0, 255).astype(np.uint8)
    axes[1, i].imshow(recon_img, cmap="gray")
    axes[1, i].axis("off")
    axes[1, i].set_title("Reconstructed", fontsize=10)

fig.suptitle(f"Original vs PCA 64D Reconstructed Images (Cumulative Explained Variance: {cumulative_var_64:.2%})",
             fontsize=16, y=0.98)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

pca_full = PCA(random_state=42).fit(X_train_scaled)
cumulative_variance = np.cumsum(pca_full.explained_variance_ratio_)

plt.figure(figsize=(12, 6))
plt.plot(
    range(1, len(cumulative_variance) + 1),
    cumulative_variance,
    marker="o", color="blue", markersize=4, linewidth=1.5
)
plt.axhline(y=0.95, color="red", linestyle="--", linewidth=2, label="95% Information Retention Threshold")
n_components_95 = np.argmax(cumulative_variance >= 0.95) + 1
plt.axvline(x=n_components_95, color="orange", linestyle=":", linewidth=2,
            label=f"~{n_components_95} PCs for 95% Information")

plt.title("Relationship Between Number of Principal Components and Cumulative Explained Variance", fontsize=16, pad=20)
plt.xlabel("Number of Principal Components", fontsize=14)
plt.ylabel("Cumulative Explained Variance Ratio (Information Retention)", fontsize=14)
plt.xlim(0, 200)
plt.ylim(0, 1.05)
plt.grid(linestyle="--", alpha=0.5)
plt.legend(fontsize=12, loc="lower right")
plt.tight_layout()
plt.show()

print(f"\nKey Conclusions")
print(f"1. When reduced to 2D: Cumulative explained variance = {cumulative_var_2d:.2%} (enough to distinguish digit clusters)")
print(f"2. When reduced to 64D: Cumulative explained variance = {cumulative_var_64:.2%} (high-quality image reconstruction)")
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)")
```