Principal Component Analysis

CMSC 173 - Machine Learning

Noel Jeffrey Pinton

October 12, 2025

Department of Computer Science
University of the Philippines - Cebu

Outline

Introduction & Motivation

Mathematical Foundations

The PCA Algorithm

PCA Variants

Evaluation & Component Selection

Applications

Best Practices & Pitfalls

Summary

Introduction & Motivation

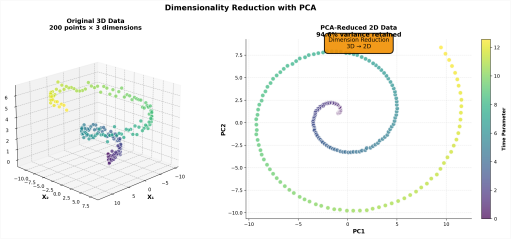
What is Principal Component Analysis?

Definition

Principal Component Analysis (PCA) is a statistical technique that transforms high-dimensional data into a lower-dimensional representation while preserving as much variance as possible.

Key Characteristics

- Unsupervised learning method
- Linear dimensionality reduction
- Orthogonal transformation
- Variance maximization



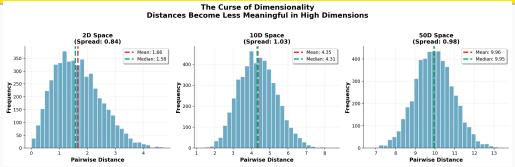
Applications:

- Data visualization
- Feature extraction
- Noise reduction
- Image compression

Goal

Find new axes (principal components) that maximize variance and are uncorrelated with each other.

The Curse of Dimensionality



Problem

As the number of features increases, data becomes increasingly sparse, making analysis difficult and computationally expensive.

Challenges in High Dimensions:

- **Data sparsity:** Points become isolated
- **Distance measures:** Lose meaning
- **Overfitting:** Models fit noise
- **Computation:** Time/memory grows exponentially

Hughes Phenomenon

Model performance initially improves with more features, then degrades beyond an optimal point.

Example Impact:

- 10 features: $10^2 = 100$ parameter pairs
- 100 features: $100^2 = 10,000$ pairs
- 1000 features: $1,000,000$ pairs!

Solution: PCA

Reduce dimensions while retaining information.

Why Dimensionality Reduction?

Motivation for PCA:

- **Visualization:** Reduce to 2D/3D for plotting
- **Storage:** Compress data efficiently
- **Speed:** Faster training and inference
- **Noise removal:** Filter out low-variance components
- **Feature extraction:** Create meaningful features
- **Collinearity:** Remove redundant features

Core Principle

Most real-world data has intrinsic dimensionality much lower than its ambient dimensionality.

Example: Images of faces

- Ambient: 10,000 pixels
- Intrinsic: 100-200 dimensions

Benefits vs Trade-offs:

Advantages

- Reduced computational cost
- Easier visualization
- Noise reduction
- Better generalization

Trade-offs

- Information loss
- Interpretability reduced
- Linear assumptions
- Sensitive to scaling

When to Use PCA

- Many correlated features
- Need for visualization
- Computational constraints
- Preprocessing for ML

Real-World Applications

Computer Vision:

- **Face recognition:** Eigenfaces
- **Image compression:** Reduce storage
- **Object detection:** Feature extraction
- **Image denoising:** Remove noise

Finance:

- Portfolio optimization
- Risk assessment
- Market trend analysis
- Fraud detection

Biology & Medicine:

- Gene expression analysis
- Medical imaging
- Drug discovery
- Disease classification

Natural Language Processing:

- Document clustering
- Topic modeling
- Sentiment analysis
- Information retrieval

Signal Processing:

- Audio compression
- Speech recognition
- Sensor data analysis
- Anomaly detection

Recommendation Systems:

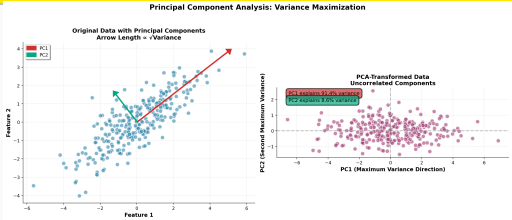
- User preference modeling
- Content filtering
- Collaborative filtering
- Feature engineering

Success Story

Netflix Prize: Teams used PCA for feature reduction and achieved significant performance gains.

Mathematical Foundations

Variance: Measuring Spread



Definition

Variance measures how far data points spread from their mean value.

For a single variable:

$$\text{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{1}$$

$$= \mathbb{E}[(X - \mathbb{E}[X])^2] \tag{2}$$

Properties:

- Always non-negative: $\text{Var}(X) \geq 0$
- Zero variance: constant values
- Units: squared original units
- **Standard deviation:** $\sigma = \sqrt{\text{Var}(X)}$

Example Calculation:

Data: $[2, 4, 6, 8, 10]$

$$\bar{x} = \frac{2+4+6+8+10}{5} = 6 \tag{3}$$

$$\text{Var}(X) = \frac{(2-6)^2 + (4-6)^2 + \cdots}{5} \tag{4}$$

$$= \frac{16+4+0+4+16}{5} = 8 \tag{5}$$

Intuition

High variance direction contains more information than low variance direction.

Covariance: Measuring Linear Relationship

Definition

Covariance measures the joint variability of two variables.

Formula:

$$\text{Cov}(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \quad (6)$$

$$= \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \quad (7)$$

Interpretation:

- $\text{Cov}(X, Y) > 0$: Positive relationship
- $\text{Cov}(X, Y) < 0$: Negative relationship
- $\text{Cov}(X, Y) = 0$: No linear relationship
- $\text{Cov}(X, X) = \text{Var}(X)$

Correlation

Normalized covariance:

$$\rho_{XY} = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \in [-1, 1]$$

Covariance Matrix:

For data $\mathbf{X} \in \mathbb{R}^{n \times d}$:

$$\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d \times d}$$

Structure:

$$\mathbf{\Sigma} = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

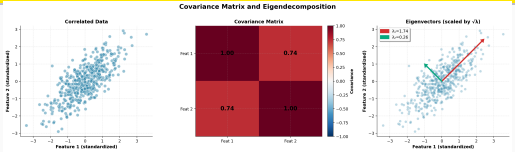
Properties:

- Symmetric: $\Sigma_{ij} = \Sigma_{ji}$
- Positive semi-definite
- Diagonal: variances
- Off-diagonal: covariances

PCA Key

Diagonalize covariance matrix to find uncorrelated components.

Eigendecomposition: The Core of PCA



Definition

For a square matrix $\mathbf{A}$, **eigendecomposition** finds vectors and scalars such that:

$$\mathbf{Av} = \lambda\mathbf{v}$$

Components:

- $\mathbf{v}$: **Eigenvector** (direction)
- λ: **Eigenvalue** (scaling factor)
- Matrix only changes *magnitude*, not direction

For Covariance Matrix:

$$\mathbf{\Sigma} = \mathbf{V\Lambda V}^T$$

where:

- $\mathbf{V}$: Eigenvector matrix
- $\mathbf{\Lambda}$: Diagonal eigenvalue matrix

Geometric Interpretation:

- Eigenvectors: Principal component directions
- Eigenvalues: Variance along each PC
- Largest eigenvalue: Most variance
- Smallest eigenvalue: Least variance

PCA Connection

- PC directions = Eigenvectors
- PC importance = Eigenvalues
- Sorted by decreasing eigenvalue

Eigendecomposition Example

Problem: Find eigenvalues and eigenvectors

$$\mathbf{\Sigma} = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix}$$

Step 1: Characteristic equation

$$\det(\mathbf{\Sigma} - \lambda \mathbf{I}) = 0$$

$$\det \begin{bmatrix} 4-\lambda & 2 \\ 2 & 3-\lambda \end{bmatrix} = 0$$

$$(4-\lambda)(3-\lambda) - 4 = 0$$

$$\lambda^2 - 7\lambda + 8 = 0$$

Step 2: Solve for eigenvalues

$$\lambda_1 = 5.56, \quad \lambda_2 = 1.44$$

Step 3: Find eigenvectors

For $\lambda_1 = 5.56$:

$$(\mathbf{\Sigma} - 5.56\mathbf{I})\mathbf{v}_1 = 0$$

$$\mathbf{v}_1 = \begin{bmatrix} 0.79 \\ 0.61 \end{bmatrix}$$

For $\lambda_2 = 1.44$:

$$\mathbf{v}_2 = \begin{bmatrix} -0.61 \\ 0.79 \end{bmatrix}$$

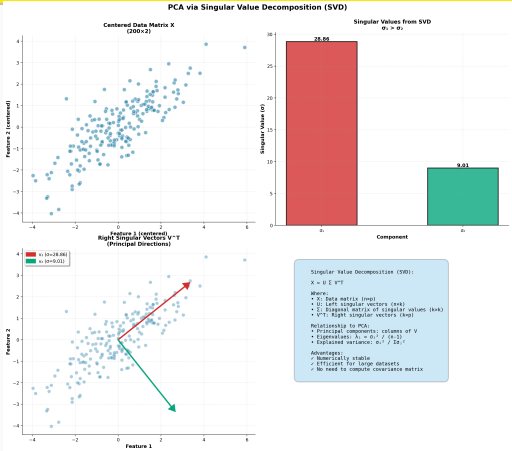
Interpretation:

- PC1 direction: $(0.79, 0.61)$
- PC1 variance: 5.56
- PC2 direction: $(-0.61, 0.79)$
- PC2 variance: 1.44
- Total variance: $5.56 + 1.44 = 7$

Note

Eigenvectors are orthogonal: $\mathbf{v}_1 \perp \mathbf{v}_2$

Singular Value Decomposition (SVD)



Definition

Any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ can be decomposed as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

Components:

- $\mathbf{U} \in \mathbb{R}^{n \times n}$: Left singular vectors

Advantages of SVD for PCA:

- More numerically stable
- Works for $n < d$ or $n > d$
- No need to form $\mathbf{X}^T\mathbf{X}$
- Efficient algorithms available

Truncated SVD

SVD vs Eigendecomposition

Eigendecomposition Approach

Steps:

1. Center data: $\tilde{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{X}}$
2. Compute covariance: $\mathbf{\Sigma} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$
3. Eigendecomposition: $\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$
4. Sort by eigenvalues

Complexity: $\mathcal{O}(d^2 n + d^3)$

Issues:

- Numerical instability for $\mathbf{X}^T\mathbf{X}$
- Squaring condition number
- Memory: storing $d \times d$ matrix

SVD Approach

Steps:

1. Center data: $\tilde{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{X}}$
2. SVD: $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
3. PCs: columns of $\mathbf{V}$
4. Variance: σ_i^2/n

Complexity: $\mathcal{O}(\min(nd^2, n^2 d))$

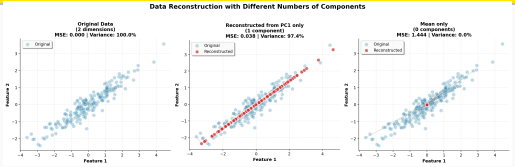
Advantages:

- Better numerical stability
- Works directly on $\mathbf{X}$
- Better for $n \ll d$ or $n \gg d$
- Modern implementations optimized

Recommendation

Use SVD for PCA in practice.

Projection and Reconstruction



Projection onto PCs:
Transform to k principal components:

$$\mathbf{Z} = \mathbf{X}\mathbf{V}_k$$

where $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ are first k PCs.

Properties:

- $\mathbf{Z} \in \mathbb{R}^{n \times k}$: Reduced representation
- Columns of $\mathbf{Z}$ are uncorrelated
- Maximum variance preserved
- Linear transformation

Reconstruction:

$$\hat{\mathbf{X}} = \mathbf{Z}\mathbf{V}_k^T = \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T$$

Example:
Original: $\mathbf{x} = [5, 3]$
PCs: $\mathbf{v}_1 = [0.8, 0.6]$
Projection:

$$z_1 = \mathbf{x}^T\mathbf{v}_1 = 5(0.8) + 3(0.6) = 5.8$$

Reconstruction:

$$\hat{\mathbf{x}} = z_1\mathbf{v}_1 = 5.8[0.8, 0.6] = [4.64, 3.48]$$

Error:

$$\|\mathbf{x} - \hat{\mathbf{x}}\| = \|[0.36, -0.48]\| = 0.6$$

Reconstruction Error

$$d$$

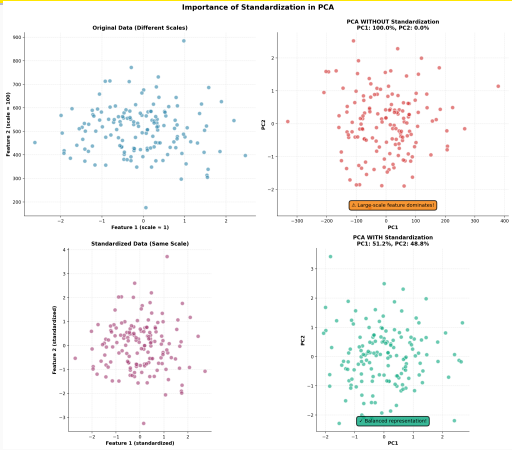
Data Centering and Standardization

Centering (Required):
Remove mean from each feature:

$$\tilde{x}_{ij} = x_{ij} - \bar{x}_j$$

Why?

- PCA finds directions of variance
- Variance computed about mean
- Ensures PC1 passes through origin



Example Impact

Without standardization:

- Income: \$20,000-\$200,000
- Age: 20-80 years
- PC1 dominated by income scale

With standardization:

The PCA Algorithm

PCA Algorithm: Step-by-Step

Algorithm 1 Principal Component Analysis

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, number of components k

Ensure: Principal components $\mathbf{V}_k$, transformed data $\mathbf{Z}$

1: **Center** the data:
2: $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$
3: $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_n \bar{\mathbf{x}}^T$
4: **Optional:** Standardize features
5: **Compute** covariance matrix:
6: $\mathbf{\Sigma} = \frac{1}{n} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$
7: **OR** compute SVD:
8: $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{D}\mathbf{V}^T$
9: **Extract** top k eigenvectors/singular vectors:
10: $\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$
11: **Project** data onto PCs:
12: $\mathbf{Z} = \tilde{\mathbf{X}}\mathbf{V}_k$
13: **return** $\mathbf{V}_k$, $\mathbf{Z}$

Key Steps Explained:

Step 1-2: Centering

Essential for computing variance correctly. Shift data so mean is at origin.

Step 3-4: Covariance/SVD

Two equivalent approaches. SVD more stable numerically.

Step 5: Top-k Selection

Sort by eigenvalues (descending) and keep largest k components.

Step 6: Projection

Transform original data to new coordinate system defined by PCs.

Complexity:

- Eigendecomposition: $\mathcal{O}(d^3)$
- SVD: $\mathcal{O}(\min(nd^2, n^2d))$

Worked Example: PCA on Toy Dataset (Part 1)

Problem Setup:
Dataset with 5 points in 2D:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 6 \end{bmatrix}$$

Goal: Find principal components

Step 1: Center the data
Compute mean:

$$\bar{\mathbf{x}} = \frac{1}{5}[15, 20]^T = [3, 4]^T$$

Centered data:

$$\tilde{\mathbf{X}} = \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

Step 2: Compute covariance matrix

$$\mathbf{\Sigma} = \frac{1}{5}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$$

$$= \frac{1}{5}\begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

$$= \frac{1}{5}\begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

Observation

Perfect correlation:
$\text{Cov}(X_1, X_2) = \text{Var}(X_1) = \text{Var}(X_2) = 2$

Worked Example: PCA on Toy Dataset (Part 2)

Step 3: Find eigenvalues

Solve: $\det(\mathbf{\Sigma} - \lambda \mathbf{I}) = 0$

$$\det \begin{bmatrix} 2-\lambda & 2 \\ 2 & 2-\lambda \end{bmatrix} = 0$$

$$(2-\lambda)^2 - 4 = 0$$

$$\lambda^2 - 4\lambda = 0$$

$$\lambda(\lambda - 4) = 0$$

Eigenvalues:

$$\lambda_1 = 4, \quad \lambda_2 = 0$$

Interpretation:

- All variance in first PC: $\lambda_1 = 4$
- Zero variance in second PC: $\lambda_2 = 0$
- Data lies on a line!

Step 4: Find eigenvectors

For $\lambda_1 = 4$:

$$(\mathbf{\Sigma} - 4\mathbf{I})\mathbf{v}_1 = 0$$

$$\begin{bmatrix} -2 & 2 \\ 2 & -2 \end{bmatrix} \mathbf{v}_1 = 0$$

Solution: $\mathbf{v}_1 = \frac{1}{\sqrt{2}}[1, 1]^T$

For $\lambda_2 = 0$:

$$\mathbf{v}_2 = \frac{1}{\sqrt{2}}[1, -1]^T$$

Verification:

$$\mathbf{v}_1^T \mathbf{v}_2 = \frac{1}{2}(1 - 1) = 0 \checkmark$$

Result

PC1: $[0.707, 0.707]$ (diagonal direction)
PC2: $[0.707, -0.707]$ (perpendicular)

Worked Example: PCA on Toy Dataset (Part 3)

Step 5: Project data onto PCs

$$\mathbf{Z} = \tilde{\mathbf{X}}\mathbf{V}$$

$$= \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{bmatrix}$$

$$= \begin{bmatrix} -2.83 & 0 \\ -1.41 & 0 \\ 0 & 0 \\ 1.41 & 0 \\ 2.83 & 0 \end{bmatrix}$$

Observation: All points lie on PC1 axis (PC2 coordinates are zero).

Step 6: Explained variance

Total variance:

$$\text{Total} = \lambda_1 + \lambda_2 = 4 + 0 = 4$$

Explained by PC1:

$$\frac{\lambda_1}{\text{Total}} = \frac{4}{4} = 100\%$$

Explained by PC2:

$$\frac{\lambda_2}{\text{Total}} = \frac{0}{4} = 0\%$$

Reconstruction (k=1):

$$\hat{\mathbf{X}} = \mathbf{Z}_{:,1}\mathbf{v}_1^T + \bar{\mathbf{X}}$$

Perfect reconstruction since all variance in PC1!

Conclusion

This toy example shows perfectly correlated data requiring only 1 dimension.

Worked Example: Non-Trivial 2D Case (Part 1)

New Dataset (4 points):

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix}$$

Step 1: Center data

Mean: $\bar{\mathbf{x}} = [2.5, 2.5]^T$

$$\tilde{\mathbf{X}} = \begin{bmatrix} -1.5 & -0.5 \\ -0.5 & -1.5 \\ 0.5 & 1.5 \\ 1.5 & 0.5 \end{bmatrix}$$

Step 2: Covariance

$$\mathbf{\Sigma} = \frac{1}{4}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$$

$$= \frac{1}{4}\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 1.25 & 0.75 \\ 0.75 & 1.25 \end{bmatrix}$$

Step 3: Eigenvalues

$$\det(\mathbf{\Sigma} - \lambda\mathbf{I}) = 0$$

$$(1.25 - \lambda)^2 - 0.5625 = 0$$

$$\lambda^2 - 2.5\lambda + 1 = 0$$

Using quadratic formula:

$$\lambda = \frac{2.5 \pm \sqrt{6.25 - 4}}{2} = \frac{2.5 \pm 1.5}{2}$$

Eigenvalues:

$$\lambda_1 = 2.0, \quad \lambda_2 = 0.5$$

Variance explained:

- PC1: $\frac{2.0}{2.5} = 80\%$
- PC2: $\frac{0.5}{2.5} = 20\%$
- Both components carry information!

Worked Example: Non-Trivial 2D Case (Part 2)

Step 4: Eigenvectors

For $\lambda_1 = 2.0$:

$$\begin{bmatrix} -0.75 & 0.75 \\ 0.75 & -0.75 \end{bmatrix} \mathbf{v}_1 = 0$$

Normalize: $\mathbf{v}_1 = \frac{1}{\sqrt{2}}[1, 1]^T = [0.707, 0.707]^T$

For $\lambda_2 = 0.5$:

$$\mathbf{v}_2 = \frac{1}{\sqrt{2}}[1, -1]^T = [0.707, -0.707]^T$$

Step 5: Transform data

$$\mathbf{Z} = \tilde{\mathbf{X}}\mathbf{V}$$

$$= \begin{bmatrix} -1.414 & -0.707 \\ -1.414 & 0.707 \\ 1.414 & -0.707 \\ 1.414 & 0.707 \end{bmatrix}$$

Step 6: Reconstruction (k=1)

Keep only PC1:

$$\hat{\tilde{\mathbf{X}}} = \mathbf{Z}_{:,1}\mathbf{v}_1^T$$

$$= \begin{bmatrix} -1.414 \\ -1.414 \\ 1.414 \\ 1.414 \end{bmatrix} \begin{bmatrix} 0.707 & 0.707 \end{bmatrix}$$

$$= \begin{bmatrix} -1.0 & -1.0 \\ -1.0 & -1.0 \\ 1.0 & 1.0 \\ 1.0 & 1.0 \end{bmatrix}$$

Reconstruction error:

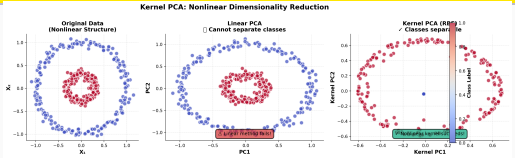
$$\text{Error} = \lambda_2 = 0.5$$

Summary

Reduced from 2D to 1D, preserving 80% of variance.

PCA Variants

Kernel PCA: Non-linear Extension



Motivation

Standard PCA finds only **linear** relationships. Many real-world datasets have non-linear structure.

Key Idea:

- Map data to high-dimensional space: $\phi : \mathbb{R}^d \to \mathcal{H}$
- Apply PCA in feature space $\mathcal{H}$
- Use kernel trick: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
- Never explicitly compute $\phi(x)$

Algorithm:

1. Compute kernel matrix: $\mathbf{K}_{ij} = K(x_i, x_j)$
2. Center kernel matrix
3. Eigendecompose: $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
4. Project: $z_i = \mathbf{U}^T \mathbf{k}(x_i)$

Common Kernels:

- **RBF:** $K(x, y) = \exp(-\gamma \|x - y\|^2)$
- **Polynomial:** $K(x, y) = (x^T y + c)^d$
- **Sigmoid:** $K(x, y) = \tanh(\alpha x^T y + c)$

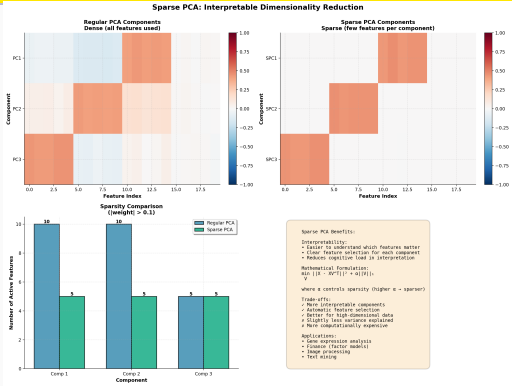
Applications

- Manifold learning
- Non-linear dimensionality reduction
- Feature extraction for non-linear data

Trade-off

More flexible but computationally expensive: $\mathcal{O}(n^3)$

Sparse PCA



Motivation

Standard PCA produces components that are linear combinations of **all** original features, making interpretation difficult.

Goal: Find PCs with **sparse loadings**

- Most loadings are exactly zero
- Only few features contribute to each PC
- Easier interpretation

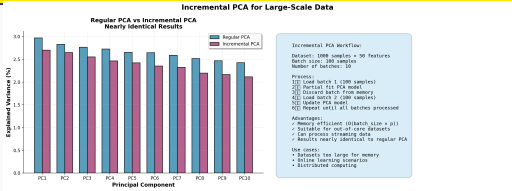
Comparison:

Method	Variance	Sparsity
Standard PCA	100%	0%
Sparse PCA	95-98%	70-90%

Applications:

- Gene expression analysis
- Financial data analysis
- Text mining
- Any domain requiring interpretability

Incremental PCA



Motivation

Standard PCA requires entire dataset in memory.
Not feasible for:

- Very large datasets
- Streaming data
- Online learning scenarios
- Limited memory systems

Key Idea: Process data in **mini-batches**:

1. Initialize with first batch
2. Update PCs incrementally with new batches
3. Maintain approximate PCs online
4. Never load full dataset

Algorithm Sketch:

Advantages:

- Memory efficient: Constant memory
- Scalable: Handles large data
- Online: Updates with new data
- Streaming: Real-time processing

Complexity:

- Per batch: $\mathcal{O}(bd^2)$ where b = batch size
- Memory: $\mathcal{O}(d^2)$ instead of $\mathcal{O}(nd)$
- Nearly identical results to batch PCA

Use Cases

- Large-scale image processing
- Sensor data streams
- Online recommendation systems

Probabilistic PCA

Motivation

Standard PCA is deterministic. Probabilistic PCA provides:

- Generative model of data
- Handling of missing values
- Uncertainty quantification
- Bayesian extensions

Model:

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_k) \tag{8}$$

$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}_d) \tag{9}$$

where:

- $\mathbf{z} \in \mathbb{R}^k$: Latent variables
- $\mathbf{W} \in \mathbb{R}^{d \times k}$: Loading matrix
- σ^2: Isotropic noise

Maximum Likelihood Solution:

$$\mathbf{W}_{ML} = \mathbf{V}_k(\mathbf{\Lambda}_k - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$$

EM Algorithm:

1: Initialize $\mathbf{W}$, σ^2
2: **repeat**
3: **E-step:** Compute $\mathbb{E}[\mathbf{z}|\mathbf{x}]$
4: **M-step:** Update $\mathbf{W}$, σ^2
5: **until** convergence

Advantages:

- Handle missing data naturally
- Mixture of PPCAs for clustering
- Bayesian inference possible
- Uncertainty quantification

Relationship to Standard PCA:

- As $\sigma^2 \to 0$: Recovers standard PCA
- Same subspace in limit
- Additional noise model

Extensions

- Factor Analysis: Non-isotropic noise
- Variational PCA: Variational inference
- Sparse PPCA: Sparse loadings

Robust PCA

Motivation

Standard PCA is sensitive to:

- Outliers: Anomalous data points
- Corrupted entries: Missing or noisy values
- Sparse errors: Localized corruption

Problem Formulation:
Decompose $\mathbf{X}$ into:

$$\mathbf{X} = \mathbf{L} + \mathbf{S}$$

where:

- $\mathbf{L}$: Low-rank (clean data)
- $\mathbf{S}$: Sparse (outliers/corruption)

Optimization:

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1$$

$$\text{subject to } \mathbf{X} = \mathbf{L} + \mathbf{S}$$

- $\|\mathbf{L}\|_*$: Nuclear norm (rank proxy)

Applications:

Video Surveillance

- $\mathbf{L}$: Static background
- $\mathbf{S}$: Moving objects
- Separates foreground/background

Face Recognition

- $\mathbf{L}$: Face structure
- $\mathbf{S}$: Occlusions, shadows
- Robust to partial occlusion

Solution Methods:

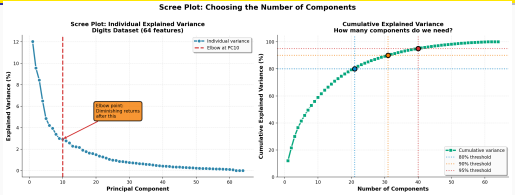
- Principal Component Pursuit (PCP)
- Alternating Direction Method (ADMM)
- Proximal gradient methods

Complexity:

- $\mathcal{O}(nd^2k)$ per iteration
- Slower than standard PCA
- Worth it for corrupted data

Evaluation & Component Selection

Scree Plot: Visualizing Variance



Definition

A **scree plot** displays eigenvalues (or explained variance) for each principal component in decreasing order.

How to Read:

- X-axis: Principal component number
- Y-axis: Eigenvalue or explained variance
- Look for "elbow" point
- Steep drop indicates important PCs
- Flat tail indicates noise

Elbow Method:

- Find point where curve bends
- Keep PCs before elbow

Example Interpretation:

- PC1: Large eigenvalue (most variance)
- PC2-3: Moderate decrease
- PC4+: Flat (noise level)
- **Decision:** Keep 3-4 components

Kaiser Criterion

Alternative rule: Keep components with eigenvalue > 1 (for correlation matrix).

Explained Variance Ratio

Individual Explained Variance:
For component i:

$$\text{EVR}_i = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

Cumulative Explained Variance:
For first k components:

$$\text{CEV}_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

Selection Criteria:

- **Fixed threshold:** $\text{CEV}_k \geq 0.95$ (95%)
- **Fixed number:** $k = 10$ components
- **Elbow method:** Visual inspection
- **Cross-validation:** Best downstream performance

Common Choice

Keep components explaining 90-95% of total variance.

Example Calculation:
Eigenvalues: $[5.2, 2.1, 0.8, 0.3, 0.1]$
Total: $\sum \lambda_i = 8.5$

PC	EVR	CEV
1	61.2%	61.2%
2	24.7%	85.9%
3	9.4%	95.3%
4	3.5%	98.8%
5	1.2%	100.0%

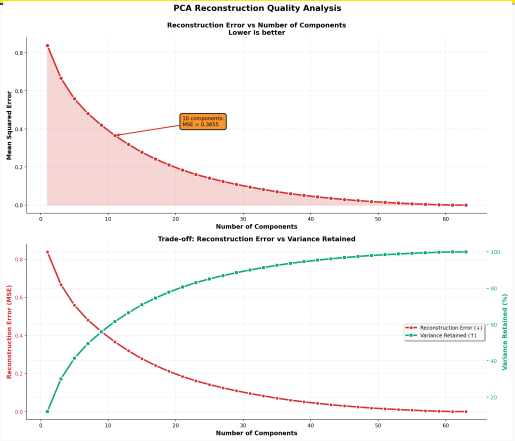
Decision:

- For 85% variance: Keep 2 PCs
- For 95% variance: Keep 3 PCs
- For 99% variance: Keep 4 PCs

Trade-off

More components: Better reconstruction, less compression

Reconstruction Error



Definition

Reconstruction error measures information lost when using only $k < d$ principal components.

Formula:

$$E_k = \|\mathbf{X} - \hat{\mathbf{X}}_k\|_F^2 = \sum_{i=k+1}^{d} \lambda_i$$

Selection Strategy:

Set error tolerance ϵ:

$$E_k \leq \epsilon \cdot \|\mathbf{X}\|_F^2$$

Choose smallest k satisfying constraint.

Example:

- Total variance: 100

Cross-Validation for Component Selection

Task-Specific Selection:

When PCA is used for downstream task:

1. Apply PCA with different k values
2. Train model on reduced data
3. Evaluate on validation set
4. Choose k with best performance

Procedure:

1: Split data: Train/Validation
2: **for** $k = 1$ to k_{max} **do**
3: Fit PCA on training set
4: Transform train & validation
5: Train classifier/regressor
6: Evaluate performance
7: **end for**
8: Select k with best validation score

Important

Fit PCA only on training data to avoid data leakage!

Example: Classification Task

k	Train Acc	Val Acc
5	0.75	0.72
10	0.82	0.79
20	0.89	0.85
50	0.95	0.84
100	0.98	0.82

Analysis:

- Best validation: $k = 20$
- $k = 50, 100$: Overfitting
- Task-specific optimum

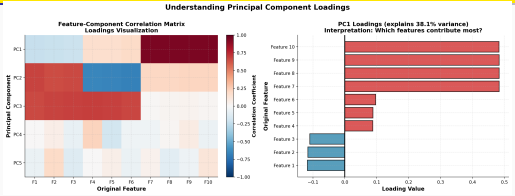
Considerations:

- Variance explained vs task performance
- Computational cost
- Interpretability needs
- Domain constraints

Best Practice

Use cross-validation when PCA is preprocessing step for supervised learning.

Component Correlation Analysis



Goal: Understand relationships between original features and principal components.

Loading Matrix:

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$$

Interpretation:

- v_{ij}: Contribution of feature j to PC i
- Large $|v_{ij}|$: Feature j important for PC i
- Sign indicates direction of contribution
- Loadings are correlations (if standardized)

Biplot:

- Visualize data and loadings together
- Arrow length: Variable importance
- Arrow direction: Correlation with PCs

Example Loading Matrix:

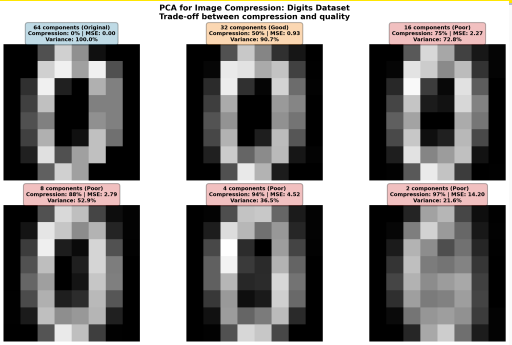
Feature	PC1	PC2	PC3
Height	0.82	0.15	-0.05
Weight	0.79	0.20	0.08
Age	0.12	0.91	0.10
Income	-0.05	0.08	0.95

Interpretation:

- PC1: "Body size" (height, weight)
- PC2: "Age" dimension
- PC3: "Income" dimension
- Clear separation of concepts

Applications

Application: Image Compression



Problem: Images contain redundant information. Can we store them more efficiently?

PCA Approach:

1. Treat each image as vector (flatten pixels)
2. Build dataset: $\mathbf{X} \in \mathbb{R}^{n \times d}$
3. Apply PCA: Find eigenvectors
4. Project: $\mathbf{Z} = \mathbf{X}\mathbf{V}_k$
5. Store: $\mathbf{Z}$ and $\mathbf{V}_k$ instead of $\mathbf{X}$

Compression Ratio:
Original: $n \times d$ values
Compressed: $n \times k + k \times d$ values

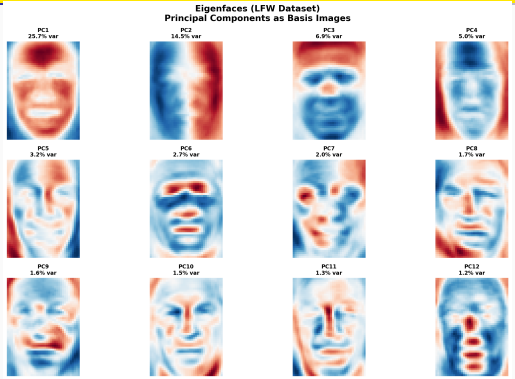
Quality vs Compression:

PCs	Compression	Quality
10	95%	Poor
50	75%	Fair
100	50%	Good
200	0%	Excellent

Trade-off

Balance between file size and visual quality. Typical choice: 80-90% variance retained.

Application: Face Recognition (Eigenfaces)



Eigenfaces Method:

1. Collect face images: n images, d pixels each
2. Apply PCA: Find "eigenfaces" (principal components)
3. Each eigenface captures facial variation
4. Represent faces in eigenface space
5. Recognition: Nearest neighbor in PC space

Advantages:

- Dimensionality reduction: $10,000 \rightarrow 100$
- Fast matching in low-dimensional space

Typical Eigenfaces:

- PC1: Average lighting
- PC2: Left-right contrast
- PC3: Facial expression
- PC4-10: Detailed features
- PC11+: Fine details/noise

Historical Note

Eigenfaces pioneered in 1991 by Turk and Pentland.

Application: Noise Filtering

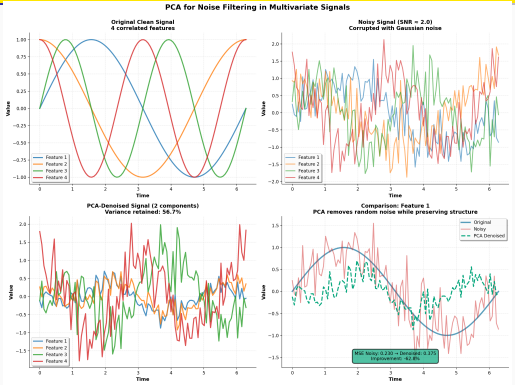
Principle:

Noise typically has:

- Low variance
- High frequency
- Random direction
- Captured by small eigenvalues

Signal has:

- High variance
- Low frequency
- Structured patterns



Example: Signal Denoising

Original signal + Gaussian noise

- Noise variance: 0.1
- PCA: Keep 5 components
- SNR improvement: 15 dB

Applications:

- Image denoising
- Audio signal processing

Application: Data Visualization

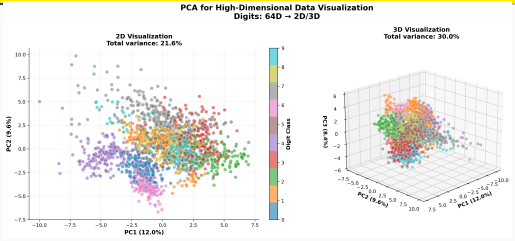
Challenge:
Cannot visualize data in $d > 3$ dimensions.

PCA Solution:
Project to 2D or 3D for visualization:

1. Apply PCA: Get principal components
2. Keep PC1 and PC2 (or PC1, PC2, PC3)
3. Plot data in reduced space
4. Preserves maximum variance

Interpretation:

- PC1 (x-axis): Direction of most variance
- PC2 (y-axis): Second most variance
- Distances approximately preserved
- Clusters may emerge



Example: Iris Dataset

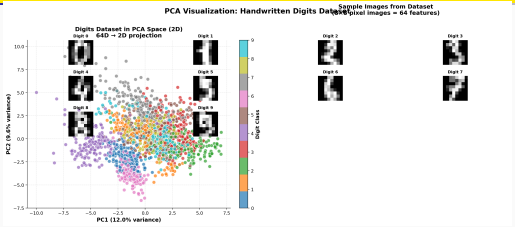
- Original: 4 dimensions
- PCA: 2D projection
- Explained variance: 95.8%
- Clear species separation visible

Comparison with Other Methods:

Method	Linear	Global
PCA	Yes	Yes
t-SNE	No	No
UMAP	No	Local
MDS	Yes/No	Yes

Limitation

Application: Exploratory Data Analysis



Digits Dataset Visualization:
High-dimensional handwritten digit images (64 dimensions) projected to 2D.

Insights from PCA:

- Digit clusters visible in PC space
- Similar digits closer together
- Outliers easily identified
- Confusion patterns apparent

Practical Workflow:

1. Load dataset
2. Standardize features
3. Apply PCA (keep 2-3 PCs)
4. Visualize with scatter plot
5. Color by labels (if available)

Observations:

- Digits 0 and 1 well-separated
- Digits 3, 5, 8 overlap slightly
- Some outliers (miswritten digits)
- PC1 and PC2 capture 30% variance

Feature Engineering

PCA projections can be used as features for downstream classification:

- Reduce 64D to 20D
- Train classifier on PC scores
- Faster training
- Often better generalization

Application: Feature Engineering

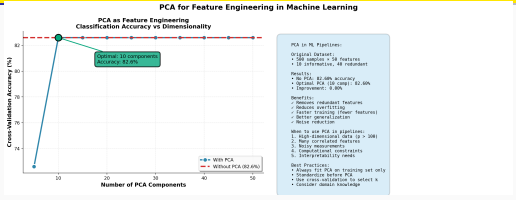
PCA as Preprocessing:
Use PCA-transformed features for ML models.

Benefits:

- **Decorrelation:** Remove multicollinearity
- **Dimensionality:** Reduce feature count
- **Noise reduction:** Filter out noisy components
- **Speed:** Faster model training
- **Regularization:** Implicit regularization effect

Pipeline:

1: Train set: Fit PCA
2: Train set: Transform with PCA
3: Test set: Transform with same PCA
4: Train classifier on PC scores
5: Evaluate on test PC scores



Example Results:

Features	Accuracy	Time
Original (1000D)	0.85	120s
PCA (100D)	0.87	15s
PCA (50D)	0.86	8s
PCA (10D)	0.79	2s

Observations:

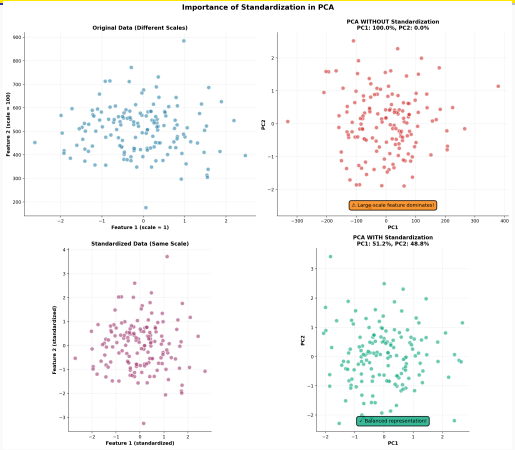
- Sweet spot: 50-100 components
- Improved accuracy with PCA
- Much faster training
- Slight degradation with too few PCs

Use Case

High-dimensional datasets where features are correlated (genomics, text, images).

Best Practices & Pitfalls

Best Practices: Standardization



Critical Decision

Should you standardize features before PCA?

Standardization:

$$z_j = \frac{x_j - \mu_j}{\sigma_j}$$

When to Standardize:

Example Impact:

Dataset: Height (cm), Weight (kg), Age (years)

Without standardization:

- Height range: 150-200
- Weight range: 50-100
- Age range: 20-70
- PC1 dominated by height

Computational Complexity

Standard PCA Complexity:

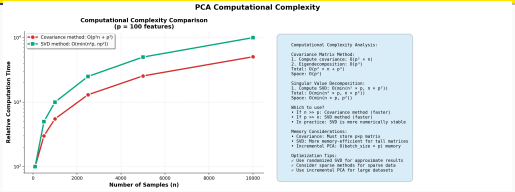
Operation	Complexity
Centering	$\mathcal{O}(nd)$
Covariance	$\mathcal{O}(nd^2)$
Eigendecomp	$\mathcal{O}(d^3)$
SVD	$\mathcal{O}(\min(nd^2, n^2d))$
Total	$\mathcal{O}(nd^2 + d^3)$

Memory Requirements:

- Data: $\mathcal{O}(nd)$
- Covariance: $\mathcal{O}(d^2)$
- Eigenvectors: $\mathcal{O}(d^2)$
- **Total:** $\mathcal{O}(nd + d^2)$

Scalability Issues:

- Large d: Covariance matrix huge
- Large n: Memory for data matrix



Solutions for Large-Scale:

Incremental PCA

- Process mini-batches
- Memory: $\mathcal{O}(bd + d^2)$
- Time: $\mathcal{O}(ndk)$

Randomized PCA

- Approximate top-k PCs
- Time: $\mathcal{O}(ndk)$
- Much faster for $k \ll d$

Sparse PCA

- Exploit data sparsity
- Reduce effective dimensionality

Common Pitfalls and How to Avoid Them

1. Data Leakage:

Mistake

Fitting PCA on entire dataset including test set.

Correct approach:

- Fit PCA only on training set
- Transform train and test separately
- Use same transformation for both

2. Forgetting to Center:

Mistake

Applying PCA without centering data.

Why it matters:

- PCA assumes zero mean
- Results will be incorrect
- Always center first!

3. Wrong Scaling Choice:

Mistake

Not standardizing when features have different scales.

Impact:

4. Interpreting PCs as Features:

Mistake

Assuming PCs have same meaning as original features.

Reality:

- PCs are linear combinations
- May not have intuitive interpretation
- Use loadings for understanding

5. Assuming Linearity:

Mistake

Applying PCA to manifold data with non-linear structure.

Solution:

- Check for non-linearity first
- Consider Kernel PCA or manifold methods

6. Ignoring Outliers:

Mistake

Not handling outliers before PCA.

Impact:

Interpreting PCA Results

What to Report:

1. **Number of components:** k chosen
2. **Explained variance:** Per component and cumulative
3. **Scree plot:** Visualize eigenvalue decay
4. **Loading matrix:** Top features per PC
5. **Biplot:** If d is small
6. **Reconstruction error:** If applicable

Interpreting Loadings:

- Examine largest magnitude loadings
- Group features by sign
- Name PC based on dominant features
- Example: "Size component", "Age component"

Example

PC1 with high loadings on [height, weight, BMI]:

- Interpretation: "Body size" component
- Positive values: Larger individuals
- Negative values: Smaller individuals

Statistical Significance:

Bootstrap approach:

- Resample data multiple times
- Compute PCA on each sample
- Check stability of components
- Report confidence intervals

Permutation test:

- Randomly permute features
- Compare eigenvalues to null distribution
- Test if variance is significant

Practical Checklist:

- ✓ Data centered/standardized?
- ✓ Scree plot shows elbow?
- ✓ Enough variance explained?
- ✓ PCs interpretable?
- ✓ No data leakage?
- ✓ Outliers addressed?
- ✓ Results validated?

Summary

Key Takeaways

Core Concepts:

- **PCA:** Linear dimensionality reduction via variance maximization
- **Principal components:** Orthogonal directions of maximum variance
- **Eigendecomposition:** Mathematical foundation
- **SVD:** Practical computation method
- **Variance explained:** Quantifies information retention

Key Steps:

1. Center (and optionally standardize) data
2. Compute covariance or apply SVD
3. Extract eigenvectors/singular vectors
4. Project data onto top-k components
5. Evaluate and interpret results

Variants:

- Kernel PCA: Non-linear extension
- Sparse PCA: Interpretable loadings
- Incremental PCA: Large-scale data
- Robust PCA: Handle outliers

Applications:

- Data visualization and exploration
- Image compression and processing
- Face recognition (eigenfaces)
- Noise filtering and denoising
- Feature engineering for ML
- Dimensionality reduction

Best Practices:

- Always center data, standardize if needed
- Use scree plot and explained variance
- Avoid data leakage in train/test split
- Check for outliers and non-linearity
- Validate component selection
- Document all preprocessing choices

Limitations:

- Assumes linear relationships
- Sensitive to outliers and scaling
- Loses interpretability
- May not preserve non-linear structure

Further Reading and Resources

Classic Papers:

- Pearson (1901): "On lines and planes of closest fit"
- Hotelling (1933): "Analysis of complex statistical variables"
- Turk & Pentland (1991): "Eigenfaces for recognition"
- Jolliffe (2002): "Principal Component Analysis" (book)

Advanced Topics:

- Independent Component Analysis (ICA)
- Non-negative Matrix Factorization (NMF)
- t-SNE and UMAP for visualization
- Autoencoders for non-linear PCA
- Gaussian Process Latent Variable Models

Software Libraries:

- `scikit-learn`: PCA, KernelPCA, IncrementalPCA
- `numpy/scipy`: Low-level linear algebra

Related Methods:

- **Linear Discriminant Analysis (LDA):** Supervised dimensionality reduction
- **Factor Analysis:** Assumes latent variables
- **Canonical Correlation Analysis:** Multi-view learning
- **Isomap:** Geodesic distances
- **Locally Linear Embedding:** Manifold learning

When to Use Alternatives:

- Non-linear structure: Kernel PCA, manifold methods
- Labeled data: LDA, supervised methods
- Visualization only: t-SNE, UMAP
- Interpretability: Sparse methods, NMF
- Very large data: Random projections, sketching

Next Steps

- Practice on real datasets
- Compare with other methods
- Explore kernel and sparse variants