

# Principal Component Analysis (PCA) – Complete Beginner-to-Advanced Revision Guide

---

## 1 . Introduction

### What is PCA?

Principal Component Analysis (PCA) is an unsupervised linear dimensionality reduction technique used to transform a high-dimensional dataset into a lower-dimensional representation while preserving as much variance as possible.

PCA creates new orthogonal features (principal components) that are linear combinations of the original features.

---

### Core Intuition

"Rotate the data to find directions of maximum variance."

- Reduce dimensionality
  - Remove redundancy (correlation)
  - Improve visualisation and efficiency
- 

### Real-World Applications

- Data visualisation ( 2 D/ 3 D)
  - Noise reduction
  - Feature extraction
  - Image compression
  - Preprocessing for ML models
- 

## 2 . Mathematical Foundations

---

### 2 . 1 Data Centering

Given dataset:

$\$ \$ X \in \mathbb{R}^{n \times d} \$ \$$

Center the data:

$\$ \$ X_c = X - \mu, \quad \mu = \frac{1}{n} \sum_{i=1}^n x_i \$ \$$

---

## 2 . 2 Covariance Matrix

Covariance matrix:

$\$ \$ \Sigma = \frac{1}{n-1} X_c^T X_c \$ \$$

---

## 2 . 3 Eigen Decomposition

Solve:

$\$ \$ \Sigma v = \lambda v \$ \$$

Where: -  $v$  = eigenvector (principal component) -  $\lambda$  = eigenvalue (variance explained)

---

## 2 . 4 Why Eigenvectors Maximise Variance

Variance along direction  $v$ :

$\$ \$ \text{Var}(Xv) = v^T \Sigma v \$ \$$

Maximise subject to  $\|v\| = 1 \rightarrow$  eigenvalue problem

---

## 2 . 5 Ordering Principal Components

Sort eigenvalues:

$\$ \$ \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \$ \$$

Top  $k$  eigenvectors form projection matrix:

$\$ \$ W_k = [v_1, v_2, \dots, v_k] \$ \$$

---

## 3 . PCA Transformation

---

### Projection to Lower Dimension

$\text{Z} = \text{X}_c \text{W}_k$

Where: -  $Z$  = reduced data

---

## 4 . SVD Formulation (Numerically Stable)

---

### Singular Value Decomposition

$\text{X}_c = \text{U} \Sigma \text{V}^T$

- Columns of  $V$  = principal directions
  - Singular values relate to variance
- 

## 5 . Explained Variance

---

### Explained Variance Ratio

$\text{EVR}_k = \frac{\lambda_k}{\sum \lambda_j}$

---

### Cumulative Explained Variance

$\sum_{i=1}^k \text{EVR}_i$

Used to select number of components

---

## 6 . Step-by-Step PCA Example

---

- 1 . Standardise data
- 2 . Compute covariance matrix
- 3 . Eigen decomposition / SVD

4 . Sort components

5 . Project data

---

## 7 . Model Evaluation

---

### Reconstruction Error

$\$ \$ |X - ZW_k^T|^2 \$ \$$

Lower error → better representation

---

## 8 . Interpretation

---

- Loadings show feature contribution
  - Components are orthogonal
  - PCA components are not directly interpretable
- 

## 9 . Assumptions

---

- Linear relationships
  - Large variance = important structure
  - Features are continuous
- 

## 1 0 . Common Pitfalls & Misconceptions

---

- X Forgetting to standardise
  - X Interpreting PCs as original features
  - X Using PCA on categorical data
  - X Assuming PCA improves accuracy
- 

## 1 1 . Python Implementation

---

### 1 1 . 1 From Scratch (NumPy)

```
import numpy as np

# Center data
X_centered = X - np.mean(X, axis=0)

# Covariance matrix
cov = np.cov(X_centered, rowvar=False)

# Eigen decomposition
eig_vals, eig_vecs = np.linalg.eigh(cov)

# Sort eigenvalues
top_idx = np.argsort(eig_vals)[::-1]
eig_vals = eig_vals[top_idx]
eig_vecs = eig_vecs[:, top_idx]

# Project data
k = 2
Z = X_centered @ eig_vecs[:, :k]
```

### 1 1 . 2 Using scikit-learn

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

X_scaled = StandardScaler().fit_transform(X)

pca = PCA(n_components=2)
Z = pca.fit_transform(X_scaled)

print(pca.explained_variance_ratio_)
```

### 1 1 . 3 Visualisation

```
import matplotlib.pyplot as plt

plt.scatter(Z[:,0], Z[:,1])
plt.xlabel('PC1')
```

```
plt.ylabel('PC2')
plt.title('PCA Projection')
plt.show()
```

---

## 1 2 . Advanced Topics

---

- Kernel PCA
  - Incremental PCA
  - Sparse PCA
  - PCA vs Autoencoders
- 

## 1 3 . When NOT to Use PCA

---

- Non-linear structure
  - Small datasets
  - Categorical data
- 

## 1 4 . Best Practices

---

- Always scale features
  - Use explained variance plot
  - Combine with domain knowledge
  - Avoid over-reduction
- 

## 1 5 . Summary

---

PCA is: - Linear - Unsupervised - Variance-preserving - Powerful for preprocessing

But: - Loses interpretability - Sensitive to scaling

---

End of Revision Guide