Principal Component Analysis (PCA)

1. Introduction to PCA

Principal Component Analysis (PCA) is a popular unsupervised linear dimensionality reduction technique used in data science and machine learning. Its primary objective is to reduce the number of features (dimensions) in a complex dataset while retaining as much of the original information (variance) as possible.

- **Definition:** Dimensionality reduction technique transforming correlated features into a smaller set of uncorrelated features called Principal Components.
- **Purpose:** Eliminates redundant features, lowers computational requirements, combats the curse of dimensionality, and enables 2D/3D visualization of complex datasets.
- **Key Principle:** It assumes linear relationships among features and prioritizes directions with maximum variance, as higher variation corresponds to more useful information.

2. Principal Component Analysis Diagram

The diagram below illustrates how PCA identifies new perpendicular axes (PC1 and PC2) aligned with the maximum variance of the original data points.

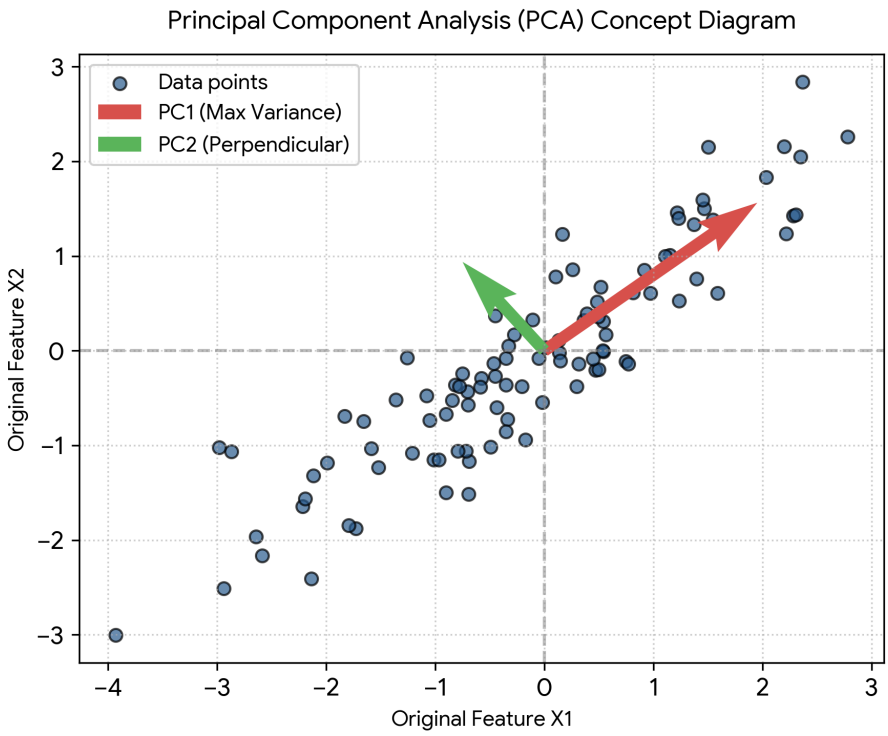


Figure 1: Transformation of original 2D data onto Principal Components (PC1 & PC2)

3. Step-by-Step Mathematical Process

Step 1: Standardize the Data

Since features may have different units and scales (e.g., Age in years vs. Salary in dollars), PCA standardizes features so each has a mean of 0 and a standard deviation of 1. This prevents high-magnitude features from dominating.

> **Standardization Formula:** $Z = (X - \mu) / \sigma$
> *where μ is the feature mean and σ is the standard deviation.*

Step 2: Calculate Covariance Matrix

The covariance matrix evaluates how features vary together (linear relationship). For features x1 and x2 across n samples:

> **Covariance Formula:** $cov(x1, x2) = \sum [(x1_i - mean(x1)) * (x2_i - mean(x2))] / (n - 1)$

Step 3: Compute Eigenvectors & Eigenvalues

PCA solves the characteristic equation of the covariance matrix matrix A:

> **Eigen Equation:** $A * X = \lambda * X$
> *where X represents the Eigenvector (direction of new axis) and λ represents the Eigenvalue (magnitude/importance of variance).*

- **Eigenvectors:** Define the principal directions (uncorrelated linear axes).
- **Eigenvalues:** Quantify the amount of variance captured along each corresponding eigenvector.

Step 4: Select Top Components & Transform Data

Sort eigenvalues in descending order. Select the top k eigenvectors corresponding to the largest k eigenvalues (e.g., retaining 95% total variance). Project original standardized data onto these k principal components.

4. Key Advantages & Disadvantages

Advantages	Disadvantages
Handles Multicollinearity by creating uncorrelated variables.	Reduced Interpretability (Principal components are linear combinations).
Reduces Noise by discarding low-variance components.	Sensitive to Feature Scaling (requires prior standardization).

Data Compression reduces memory and computational cost.	Potential Information Loss if too few components are selected.
Enables Easy Visualization of high-dimensional data in 2D/3D.	Assumes Linearity (struggles with non-linear structures).

5. Python Implementation Reference

Below is a clean workflow for applying PCA using scikit-learn:

> **Python Code Snippet:**
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# 1. Sample Data
data = {'Height': [170, 165, 180, 175, 160],
        'Weight': [65, 59, 75, 68, 55],
        'Age': [30, 25, 35, 28, 22]}
df = pd.DataFrame(data)

# 2. Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# 3. Apply PCA (reduce from 3 to 2 components)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```