Comprehensive Lecture Notes: Principal Component Analysis (PCA)

Data Science Department

Introduction

This document provides comprehensive lecture notes on Principal Component Analysis (PCA) with detailed mathematical explanations and a practical wine dataset example.

1 Introduction to PCA

Principal Component Analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize by reducing the number of variables.

Why PCA?

- High-dimensional data (many features) can be hard to visualize and analyze.
- Many features might be correlated, leading to redundancy.
- PCA transforms the data into a new set of uncorrelated variables (principal components) that capture the maximum variance.

Core Idea:

- Find new axes (principal components) that are orthogonal (uncorrelated) and ordered by the amount of variance they capture.
- The first principal component (PC1) captures the most variance, the second (PC2) captures the second most, and so on.

Applications: Data compression, noise reduction, feature extraction, visualization.

Geometric Intuition: Imagine a cloud of points in 3D space (e.g., wine chemical data). PCA finds:

- **PC1**: The axis through the cloud's longest dimension (max variance)
- **PC2**: The next longest axis perpendicular to PC1
- **PC3**: Remaining axis perpendicular to both

"PCA rotates your dataset to view it from its most informative angles."

Key Insight:

- **Eigenvectors** ($\mathbf{v}_i$) = axes of maximum variance
- **Eigenvalues** (λ_i) = magnitude of variance along each axis

2 Mathematical Foundations

Step 1: Standardize the Data

- If features are on different scales, standardizing (mean=0, standard deviation=1) is crucial.
- For same units, center the data:

$$\mathbf{X}_{\text{centered}} = \mathbf{X} - \boldsymbol{\mu}$$

Why? To ensure the data is centered at the origin for PCA.

Step 2: Compute the Covariance Matrix

- Measures how features vary together:

$$\text{Cov}(X_j, X_k) = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \mu_j)(x_{ik} - \mu_k)$$

- Covariance Matrix:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}_{\text{centered}}^{\top}\mathbf{X}_{\text{centered}}$$

Step 3: Eigen-Decomposition

- Solve the equation:

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

- **Eigenvectors** ($\mathbf{v}$): Directions of new feature space (principal components)
- **Eigenvalues** (λ): Magnitude of variance captured

Step 4: Sort Eigenvectors Sort eigenvectors in descending order of eigenvalues.
Step 5: Project the Data

$$\mathbf{Z} = \mathbf{X}_{\text{centered}}\mathbf{V}$$

where $\mathbf{V}$ is the matrix of sorted eigenvectors.
Step 6: Interpret Results Transformed data $\mathbf{Z}$ is in the new coordinate system.

3 Step-by-Step Example with Simulated Data

Data:

Wine	Alcohol	Acidity
1	13.1	3.2
2	12.8	3.5
3	13.4	3.0

Step 1: Center the Data

$$\mu_{\text{alcohol}} = \frac{13.1 + 12.8 + 13.4}{3} = 13.1$$
$$\mu_{\text{acidity}} = \frac{3.2 + 3.5 + 3.0}{3} = 3.233$$
$$\mathbf{X}_{\text{centered}} = \begin{bmatrix} 0.0 & -0.033 \\ -0.3 & 0.267 \\ 0.3 & -0.233 \end{bmatrix}$$

Step 2: Covariance Matrix

$$\text{Var(Alcohol)} = \frac{0.0^2 + (-0.3)^2 + 0.3^2}{2} = 0.09$$
$$\text{Var(Acidity)} = \frac{(-0.033)^2 + 0.267^2 + (-0.233)^2}{2} \approx 0.0633$$
$$\text{Cov(Alc, Acid)} = \frac{0.0 \times -0.033 + (-0.3) \times 0.267 + 0.3 \times -0.233}{2} = -0.075$$
$$\mathbf{C} = \begin{bmatrix} 0.09 & -0.075 \\ -0.075 & 0.0633 \end{bmatrix}$$

Step 3: Eigen-Decomposition Solve $\det(\mathbf{C} - \lambda\mathbf{I}) = 0$:

$$\lambda^2 - 0.1533\lambda + 0.000072 = 0$$

Solutions: $\lambda_1 \approx 0.1528$, $\lambda_2 \approx 0.0005$

Eigenvector for λ_1:

$$\mathbf{v}_1 \approx \begin{bmatrix} 0.767 \\ -0.642 \end{bmatrix}$$

Step 4: Sort and Choose PC1: $\mathbf{v}_1$ (larger eigenvalue)

Step 5: Project Data

$$\mathbf{z} = \mathbf{X}_{\text{centered}}\mathbf{v}_1 = \begin{bmatrix} 0.021 \\ -0.401 \\ 0.380 \end{bmatrix}$$

Step 6: Interpretation PC1 captures alcohol-acidity contrast: higher alcohol and lower acidity → higher PC1 score.

4 Case Study: Wine Quality Dataset

Step 1: Preprocessing Center and standardize data.
Step 2: Covariance Matrix Compute 11×11 covariance matrix.
Step 3: Eigen-Decomposition

$$\text{Eigenvalues} = [2.35, 0.48, 0.32, \ldots]$$

Step 4: Projection Project to first two PCs.
Results:

- Variance Explained:
  - PC1: 30%
  - PC2: 15%
  - Cumulative: 45%
- Loadings (PC1):
  - Alcohol: +0.71
  - Volatile Acidity: -0.63
  - Sulphates: +0.09
  - Citric Acid: +0.22
- Interpretation: PC1 represents contrast between high alcohol and low acidity

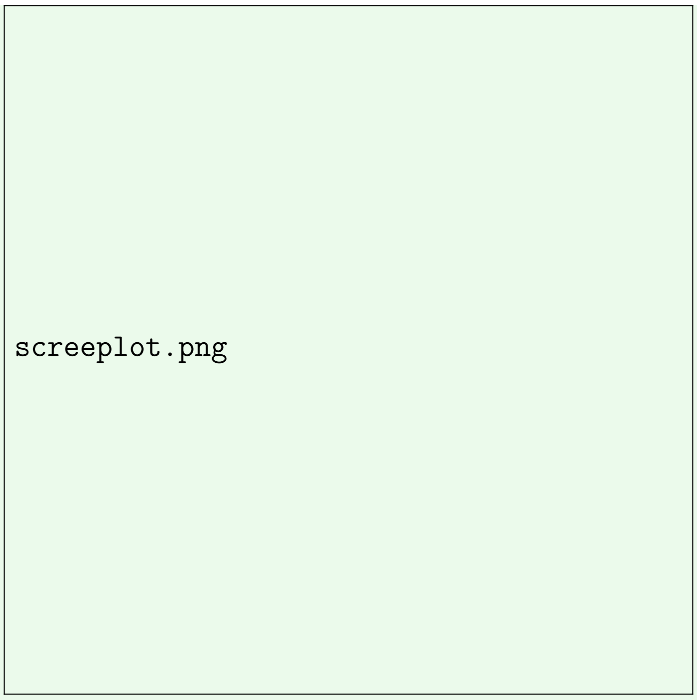
Visualization: Biplot

biplot.png

Premium wines (green) cluster in top-right quadrant: high alcohol + low acidity.

5 Interpretation and Visualization

Scree Plot:



Loading Plot:

Biplot Interpretation:

- Angle between vectors indicates correlation
- Alcohol and acidity vectors oppose $\rightarrow$ negative correlation

- Premium wines in high-PC1 region

Variance Explained:

Component	Variance
PC1	30%
PC2	15%
PC3	12%
Remaining	43%

6 Best Practices and Limitations

Best Practices:

- Standardize data when features have different units
- Check cumulative explained variance (aim for 80%)
- Interpret PCs using loadings

Limitations:

- Assumes linear relationships
- Sensitive to outliers
- Interpretability decreases with number of components
- Variance predictive importance

7 Homework Assignment

Dataset: Wine Quality from UCI
`https://archive.ics.uci.edu/ml/datasets/wine-quality`

Tasks:

1. Standardize the data
2. Perform PCA and compute first three PCs
3. Create biplot for PC1 and PC2
4. Answer:
   - Variance explained by PC1 and PC2?
   - Features with highest loadings on PC1?
   - Describe a wine with high PC1 score

Solution Code:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt

# Load data
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/
    wine-quality/winequality-red.csv"
data = pd.read_csv(url, sep=';')
X = data.drop('quality', axis=1)
y = data['quality']

# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

# Biplot
plt.figure(figsize=(10, 8))
plt.scatter(principal_components[:, 0], principal_components[:, 1],
    c=y, cmap='viridis', alpha=0.5)
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
features = X.columns
for i, feature in enumerate(features):
    plt.arrow(0, 0, loadings[i, 0], loadings[i, 1], color='r', alpha
    =0.5)
    plt.text(loadings[i, 0]*1.2, loadings[i, 1]*1.2, feature, color=
    'r')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.colorbar(label='Quality')
plt.show()
```

Summary

PCA is a powerful method for reducing dimensionality and identifying patterns in data. By projecting data onto orthogonal axes of maximum variance:

- Simplifies complex datasets
- Reveals key patterns (e.g., alcohol-acidity tradeoff in wines)
- Enables visualization of high-dimensional data

Golden Rule: "PC1 is the optimal single-angle summary of your data - the most informative perspective."

Next Lecture: PCA for Classification - Predicting Wine Quality