pca_from_scratch

August 21, 2025

1 PCA (Principal Component Analysis) — From Scratch

This notebook derives PCA, implements it from scratch (centering, covariance, eigen-decomposition), and demonstrates dimensionality reduction, reconstruction, and explained variance on synthetic data.

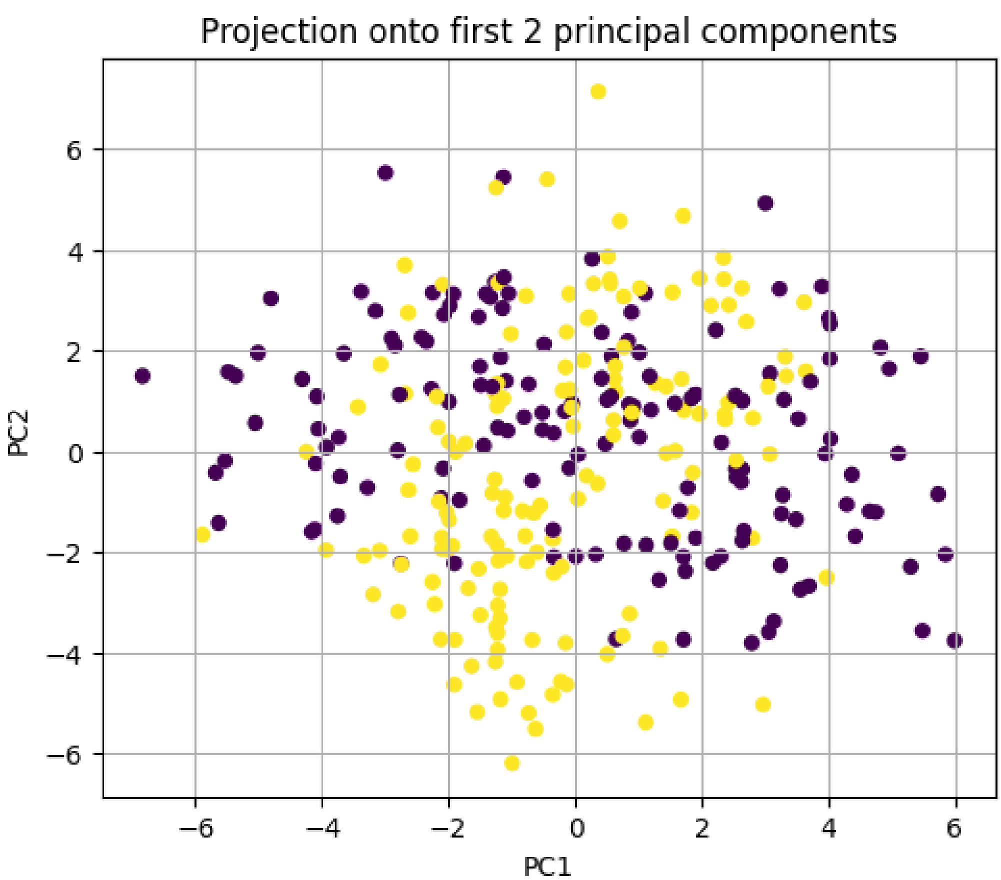
1.1 1. Theory

PCA finds orthogonal directions (principal components) that maximize data variance. For centered data matrix X (shape $n \times d$), the covariance matrix is $C = \frac{1}{n-1} X^T X$. Eigen-decomposition of C yields eigenvalues and eigenvectors. Project onto top-k eigenvectors to reduce dimensionality.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
np.random.seed(0)

def pca_from_scratch(X, k=None, return_details=False):
    # X: (n_samples, n_features)
    Xc = X - X.mean(axis=0)
    n = Xc.shape[0]
    # covariance matrix
    C = (Xc.T @ Xc) / (n - 1)
    # eigen-decomposition (symmetric -> use eigh)
    eigvals, eigvecs = np.linalg.eigh(C)  # ascending order
    # sort descending
    idx = np.argsort(eigvals)[::-1]
    eigvals = eigvals[idx]
    eigvecs = eigvecs[:, idx]
    if k is None:
        k = X.shape[1]
    components = eigvecs[:, :k]
    projected = Xc @ components
    reconstructed = projected @ components.T + X.mean(axis=0)
    if return_details:
        explained_variance_ratio = eigvals / eigvals.sum()
        return projected, reconstructed, eigvals, components,␣
↪explained_variance_ratio
```

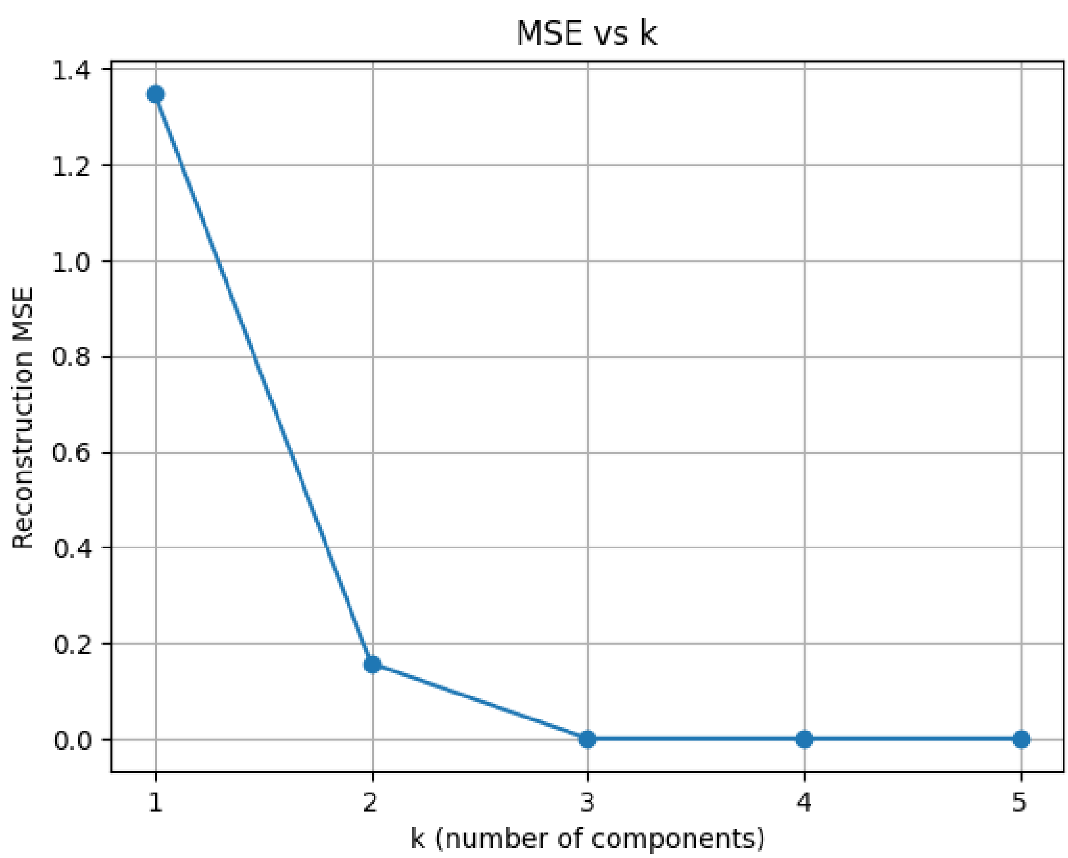
```
    return projected, reconstructed

# Example: synthetic data with 3 informative features, visualize 2D projection
X, y = make_classification(n_samples=300, n_features=5, n_informative=3,␣
 ↪random_state=0)
proj, rec, eigvals, comps, evr = pca_from_scratch(X, k=2, return_details=True)
print('Top eigenvalues:', eigvals[:5])
print('Explained variance ratio (top 5):', evr[:5])

plt.figure(figsize=(6,5))
plt.scatter(proj[:,0], proj[:,1], c=y, cmap='viridis', s=25)
plt.xlabel('PC1'); plt.ylabel('PC2'); plt.title('Projection onto first 2␣
 ↪principal components')
plt.grid(True)
plt.show()
```

```
Top eigenvalues: [ 6.30086639e+00  5.97971070e+00  7.89767074e-01
8.97138891e-16
 -4.53049681e-16]
Explained variance ratio (top 5): [ 4.82073487e-01  4.57502161e-01
6.04243518e-02  6.86392707e-17
 -3.46624140e-17]
```



1.1.1 Reconstruction error vs number of components

[2]:
```
# Reconstruction MSE as a function of k
Xc = X - X.mean(axis=0)
_, _, eigvals_all, eigvecs_all, _ = pca_from_scratch(X, k=None,␣
 ↪return_details=True)
mses = []
ks = list(range(1, X.shape[1]+1))
for k in ks:
    proj_k, rec_k = pca_from_scratch(X, k=k)[:2]
    mse = np.mean((X - rec_k)**2)
    mses.append(mse)

plt.plot(ks, mses, marker='o')
plt.xlabel('k (number of components)'); plt.ylabel('Reconstruction MSE'); plt.
 ↪title('MSE vs k')
plt.xticks(ks)
plt.grid(True)
plt.show()
```



1.2 2. Notes

- We used covariance-eigendecomposition (good when d is not huge).
- For large d or numerical stability, using SVD on the centered data is recommended.