Lecture 9: Dimensionality Reduction & Feature Extraction

Luu Minh Sao Khue

From PCA and SVD to manifold learning — understanding variance, eigen decomposition, projection, whitening, and nonlinear embedding methods like t-SNE and UMAP.

1. Learning Objectives

After this lecture, you will be able to:

- Explain the intuition and derivation of Principal Component Analysis (PCA).
- Compute eigenvalues and eigenvectors of a covariance matrix.
- Understand Singular Value Decomposition (SVD) and its link to PCA.
- Perform data whitening and feature projection.
- Describe t-SNE and UMAP for nonlinear dimensionality reduction.
- Implement PCA in Python and visualize 2D/3D projections.

2. Intuition

High-dimensional data often contain redundancy. Many features are correlated or carry little new information. **Dimensionality reduction** seeks to represent data with fewer features while preserving as much structure or variance as possible.

- **Feature Extraction:** Create new features (linear or nonlinear) summarizing the original ones (e.g., PCA components).
- **Feature Selection:** Choose a subset of existing features.

Dimensionality reduction is critical for:

- Visualization (2D/3D projection of high-dimensional data)
- Noise reduction and decorrelation
- Improving model generalization and computation speed

3. Principal Component Analysis (PCA)

3.1 Goal

Given data matrix $X \in \mathbb{R}^{n \times p}$ (rows = samples, columns = features), PCA finds orthogonal directions (principal components) that maximize variance:

$$\max_{w_1,\ldots,w_k} \sum_{i=1}^{k} \mathrm{Var}(Xw_i) \quad \text{s.t.} \quad w_i^T w_j = \delta_{ij}.$$

3.2 Covariance Matrix

Center data:

$$\tilde{X} = X - \mathbf{1}\bar{x}^T, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Compute covariance:

$$\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}.$$

3.3 Eigen Decomposition

Find eigenvalues λ_j and eigenvectors v_j:

$$\Sigma v_j = \lambda_j v_j.$$

Eigenvectors v_j define orthogonal directions in feature space; eigenvalues λ_j measure variance along them.

3.4 Explained Variance

$$\text{Explained variance ratio} = \frac{\lambda_j}{\sum_i \lambda_i}.$$

Usually, we choose k such that cumulative explained variance $\geq 90\%$.

3.5 Projection

Project centered data to first k components:

$$Z = \tilde{X}V_k,$$

where $V_k = [v_1, \ldots, v_k]$. Each row of Z is a k-dimensional representation of the sample.

3.6 Whitening (Optional)

To make projected features uncorrelated and unit variance:

$$Z_{\text{white}} = \Lambda^{-\frac{1}{2}} V^T \tilde{X}.$$

Here $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$.

4. Singular Value Decomposition (SVD)

SVD is a matrix factorization:

$$\tilde{X} = USV^T,$$

where:

- $U \in \mathbb{R}^{n \times n}$: orthogonal (left singular vectors)
- $S \in \mathbb{R}^{n \times p}$: diagonal with singular values s_i
- $V \in \mathbb{R}^{p \times p}$: orthogonal (right singular vectors)

Relationship to PCA:

$$\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X} = V\left(\frac{S^TS}{n-1}\right)V^T.$$

Thus, eigenvectors of Σ are columns of V, and eigenvalues are $\lambda_i = s_i^2/(n-1)$.

5. Geometric Interpretation

PCA rotates the coordinate system so that:

- The first axis (PC1) aligns with the direction of maximum variance.
- Each next axis is orthogonal to the previous and captures the next largest variance.

Data are projected onto these new axes — an orthogonal basis capturing most variability in fewer dimensions.

6. Manifold Learning

PCA is linear — it assumes the data lie near a linear subspace. **Manifold learning** handles nonlinear structures, assuming data lie on a low-dimensional curved manifold embedded in high dimensions.

6.1 t-SNE (t-Distributed Stochastic Neighbor Embedding)

- Models pairwise similarities in high- and low-dimensional spaces.
- Converts distances into probabilities of neighborhood similarity.

For high-dimensional data:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)},$$

symmetrized as

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}.$$

For low-dimensional embeddings y_i, similarities are:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}.$$

t-SNE minimizes the Kullback–Leibler (KL) divergence:

$$C = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Interpretation: Nearby points in high-dimensional space remain close in the embedding. Distances between far points are not preserved; t-SNE is good for visualizing local clusters.

6.2 UMAP (Uniform Manifold Approximation and Projection)

UMAP builds on manifold and topological theory:

- Constructs a weighted graph of nearest neighbors in high dimensions.
- Optimizes low-dimensional representation preserving the fuzzy topological structure.

Formally, UMAP:

1. Builds a high-dimensional graph G_H where edges reflect probability of neighbor connection based on distance and local density.
2. Builds a low-dimensional graph G_L and minimizes cross-entropy:

$$C = \sum_{(i,j)} w_{ij} \log \frac{w_{ij}}{w'_{ij}} + (1 - w_{ij}) \log \frac{1 - w_{ij}}{1 - w'_{ij}}.$$

UMAP preserves both local and some global structures, scales better than t-SNE, and allows transform of new data (parametric form).

7. Comparison of Dimensionality Reduction Methods

Method	Type	Preserves	Use Case
PCA	Linear	Global variance	Preprocessing, compression
t-SNE	Nonlinear	Local structure	Visualization
UMAP	Nonlinear	Local + global	Visualization, clustering

8. Python Implementation: PCA, t-SNE, UMAP

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris, make_blobs
from sklearn.preprocessing import StandardScaler
```

```python
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap

# ==== Dataset 1: Iris ====
iris = load_iris()
X = StandardScaler().fit_transform(iris.data)
y = iris.target

# ==== PCA ====
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
print("Explained variance ratio:", pca.explained_variance_ratio_)

plt.figure(figsize=(6,5))
plt.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap='viridis')
plt.title("Iris PCA Projection")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()

# ==== Dataset 2: Synthetic 3D Gaussian ====
X_blob, y_blob = make_blobs(n_samples=300, centers=4, n_features=3,
                            cluster_std=1.2, random_state=42)
X_blob = StandardScaler().fit_transform(X_blob)

# PCA to 2D
pca3 = PCA(n_components=2).fit(X_blob)
Z = pca3.transform(X_blob)

# t-SNE
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
Z_tsne = tsne.fit_transform(X_blob)

# UMAP
umap_model = umap.UMAP(n_neighbors=15, min_dist=0.1, random_state=42)
Z_umap = umap_model.fit_transform(X_blob)

# Visualization
fig, axes = plt.subplots(1,3, figsize=(15,4))
axes[0].scatter(Z[:,0], Z[:,1], c=y_blob, cmap='tab10')
axes[0].set_title("PCA 2D")

axes[1].scatter(Z_tsne[:,0], Z_tsne[:,1], c=y_blob, cmap='tab10')
axes[1].set_title("t-SNE 2D")

axes[2].scatter(Z_umap[:,0], Z_umap[:,1], c=y_blob, cmap='tab10')
axes[2].set_title("UMAP 2D")

plt.tight_layout()
plt.show()
```

9. Practical Tips

- Always center and scale features before PCA.
- Use scree plot (eigenvalues vs. components) to decide k.
- PCA is deterministic; t-SNE/UMAP are stochastic — set random seed.
- t-SNE: good for cluster visualization, not for downstream models.
- UMAP: faster, can embed new samples, preserves more global structure.
- Whitening helps when features have very different variances.

10. Summary

- PCA finds orthogonal directions of maximal variance using eigen decomposition of the covariance matrix.
- SVD provides a numerically stable implementation of PCA.
- Whitening decorrelates features to unit variance.
- t-SNE preserves local neighborhoods using probabilistic similarities and KL divergence minimization.
- UMAP builds a topological graph and minimizes cross-entropy to preserve both local and global structure.
- Dimensionality reduction aids visualization, denoising, and computational efficiency.

11. Exercises

1. Derive that PCA directions maximize variance subject to orthogonality.
2. Compute covariance and perform PCA manually on a small dataset.
3. Plot scree plot and cumulative explained variance curve.
4. Compare t-SNE and UMAP embeddings for a dataset of your choice.
5. Implement whitening transformation and verify feature correlation.
6. Visualize PCA vs. t-SNE vs. UMAP and interpret differences.