Principal Component Analysis (PCA)

Motivating Examples

- Images of digits (e.g., MNIST): Each image is 784-dimensional
- Sensors on wearables: 10s–100s of channels, high redundancy
- Environmental data: temperature, humidity, air quality across locations

Motivating Examples

- Images of digits (e.g., MNIST): Each image is 784-dimensional
- Sensors on wearables: 10s–100s of channels, high redundancy
- Environmental data: temperature, humidity, air quality across locations

Goal: Reduce dimensions while preserving key structure

Why PCA? Intuition

- Data lives in high dimensions but often varies in a low-dimensional subspace
- PCA finds new axes (principal directions) capturing maximum variance
- Reduces noise, saves space, helps visualize

Why PCA? Intuition

- Data lives in high dimensions but often varies in a low-dimensional subspace
- PCA finds new axes (principal directions) capturing maximum variance
- Reduces noise, saves space, helps visualize

Key Idea
Project data onto top-k directions of highest variance

Visual Intuition: Elliptical Gaussians

pca_ellipse.png

Minimal Math

Given: Data matrix $X \in \mathbb{R}^{n \times d}$

1. Center the data: $\mathbf{X}_{\text{centered}} = X - \mu$
2. Compute covariance:

$$\Sigma = \frac{1}{n} X^\top X$$

3. Eigendecompose:

$$\Sigma = U \Lambda U^\top$$

4. Project onto top-k components:

$$Z = X U_k$$

Minimal Math

Given: Data matrix $X \in \mathbb{R}^{n \times d}$

1. Center the data: $\mathbf{X}_{\text{centered}} = X - \mu$
2. Compute covariance:

$$\Sigma = \frac{1}{n} X^\top X$$

3. Eigendecompose:

$$\Sigma = U \Lambda U^\top$$

4. Project onto top-k components:

$$Z = X U_k$$

Reconstruction: $\hat{X} = Z U_k^\top + \mu$

In Code (PyTorch)

- Center data: `X_centered = X - X.mean(0)`
- Covariance: `cov = X_centered.T @ X_centered / N`
- Eigenvectors: `eigvals, eigvecs = torch.linalg.eigh(cov)`
- Project: `X_proj = (X @ eigvecs[:, -k:])`

Example: MNIST Digits

mnist_pca_recon.png

Summary

- PCA finds orthogonal directions of max variance
- Works via eigendecomposition of the covariance
- Useful for compression, denoising, visualization

Summary

- PCA finds orthogonal directions of max variance
- Works via eigendecomposition of the covariance
- Useful for compression, denoising, visualization

Next: PCA for downstream ML tasks