Lecture 4.4: Dimensional reduction

谢丹
清华大学数学系

October 20, 2025

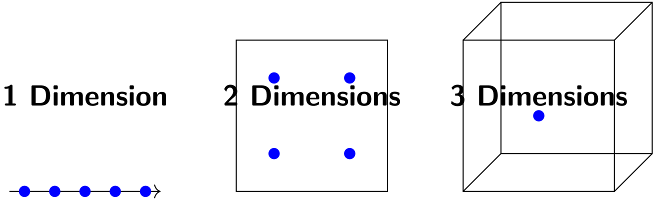
What is the Curse of Dimensionality?

Simple Definition

Problems that happen when we have too many features (dimensions) in our data.

- Data becomes very **sparse** (spread out)
- Distances between points become **meaningless**
- Need **much more data** to learn patterns
- Algorithms work **worse**, not better

Visual Example: Data Becomes Sparse



Good coverage | Some gaps | Very sparse

Key Insight

Each new dimension makes data exponentially more spread out!

The Distance Problem

2D: Good distances **100D: Similar distances**

What Happens

In high dimensions, all points become nearly the same distance apart.
This breaks algorithms like K-Nearest Neighbors!

How Much Data Do We Need?

Dimensions	Points for 10% coverage
1D	10
2D	100
3D	1,000
10D	10,000,000,000
20D	100,000,000,000,000,000

Exponential Growth!

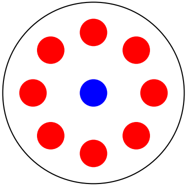
- 10D: 10 billion points needed
- 20D: 100 million billion points
- More than all the data in the world!

When High Dimensions Help

The Blessing of Dimensionality

Sometimes more dimensions can be good!

- **SVMs**: Can find better separation boundaries
- **Deep Learning**: Can learn complex patterns
- **Privacy**: Harder to identify individuals



Not separable in 2D, but separable in 3D!

Summary: Key Points

- **Curse**: High dimensions make data sparse and distances useless
- **Problem**: Need exponentially more data, algorithms break
- **Solution**: Use fewer features, choose algorithms carefully
- **Remember**: Quality over quantity for features!

Final Thought

Don't just add more features - think about whether they actually help!

The Curse of Dimensionality

- Real-world datasets often have **many features** (high dimensionality).
- This can cause problems for analysis and visualization.

We will discuss following two methods for dimensional reduction.

- **PCA** is an (unsupervised) powerful technique for dimensionality reduction.
  - It simplifies complexity.
  - It reveals hidden patterns.
- **LDA** is a (supervised) for dimensionality reduction.

What is Principal Component Analysis?

Definition

PCA is an **unsupervised** statistical method that transforms the original variables into a new set of variables called Principal Components (PCs).

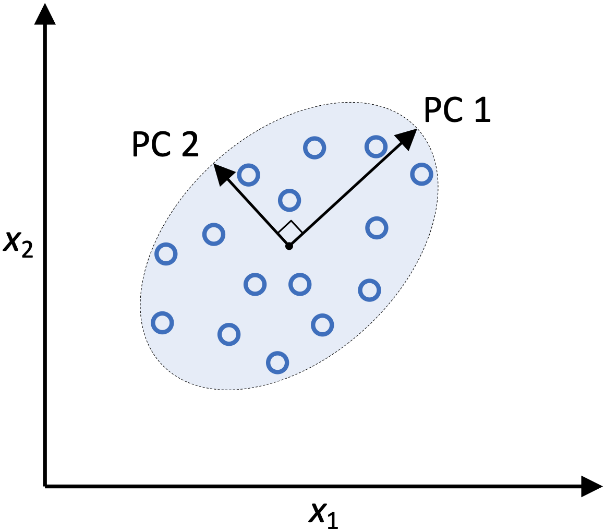
- PCs are **linear combinations** of the original variables.
- They are **orthogonal** (uncorrelated) to each other.
- They are ordered so that the first few retain most of the variation present in the original dataset.

Goal: Project data onto a lower-dimensional subspace that captures most of the variance.

The Intuition Behind PCA

Finding a New Perspective
Find the "best" angle to view the data to see the maximum spread.

1. **PC1**: Direction of maximum variance.
2. **PC2**: Orthogonal direction with next highest variance.


PC 1
PC 2
x_2
x_1

Step 1: Standardize the Data

Given a dataset with m samples and n features: $\mathbf{X} = [x_1, x_2, ..., x_n]$ (a $m \times n$ matrix)

$$\text{Center the data:} \quad z_{ij} = x_{ij} - \mu_j$$
$$\text{where } \mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$

Why? To ensure all features have a mean of zero, preventing features with large scales from dominating the analysis.

Step 2 & 3: Covariance Matrix and Eigendecomposition

Step 2: Compute the Covariance Matrix

$$\mathbf{C} = \frac{1}{m-1}\mathbf{Z}^T\mathbf{Z}$$

The covariance matrix **C** shows how features vary with each other.

Step 3: Perform Eigendecomposition

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

- $\mathbf{v}_i$: **Eigenvectors** (the Principal Components). They give the direction.
- λ_i: **Eigenvalues**. They give the magnitude (amount of variance explained).

Step 4 & 5: Selection and Projection

Step 4: Sort and Select

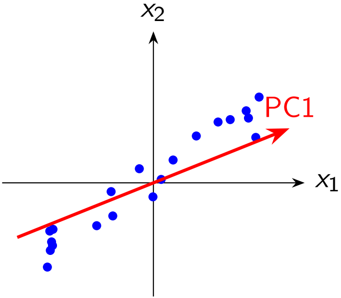
- Sort eigenvectors by their eigenvalues in descending order: $\lambda_1 > \lambda_2 > ... > \lambda_n$.
- Choose the top k eigenvectors that capture sufficient variance (e.g., 95%).

Step 5: Project the Data

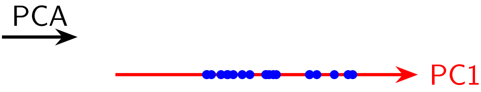
$$\mathbf{T} = \mathbf{Z} \cdot \mathbf{W}_k$$

- $\mathbf{W}_k$: The **projection matrix** formed from the top k eigenvectors.
- $\mathbf{T}$: The transformed data in the new k-dimensional space. ($m \times k$ matrix)

2D Data



1D Projection



Dimension reduced from 2D to 1D

Summary

- PCA is a linear dimensionality reduction technique.
- It finds new, uncorrelated features (PCs) that are ordered by variance.
- The process involves standardization, covariance calculation, and eigendecomposition.
- It is widely used for **visualization**, **noise reduction**, and **feature extraction**.
- **Limitation**: It assumes linear relationships and that high variance implies high importance.

We can then use the projected data to do the analysis, i.e. regression and classification.

What is Linear Discriminant Analysis?

Definition

LDA is a **supervised** dimensionality reduction technique that finds a linear combination of features that best separates two or more classes of objects or events.

- **Supervised**: Uses class labels during training
- **Linear**: Finds linear boundaries between classes
- **Discriminant**: Maximizes class separability

Goal: Project data to maximize class separation

LDA Model Assumptions

Key Assumptions

1. **Class-conditional distributions are Gaussian:**

$$P(\mathbf{x}|y = k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

2. **Shared covariance matrix:** All classes have same $\boldsymbol{\Sigma}$
3. **Class priors:** $P(y = k) = \pi_k$, with $\sum_{k=1}^{K} \pi_k = 1$

Parameters to Estimate

- Class means: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
- Common covariance: $\boldsymbol{\Sigma}$
- Class priors: $\pi_1, \ldots, \pi_K$

Key Insight for Dimensionality Reduction I

Fundamental Observation

Only the relative distance to class centers matters for classification.

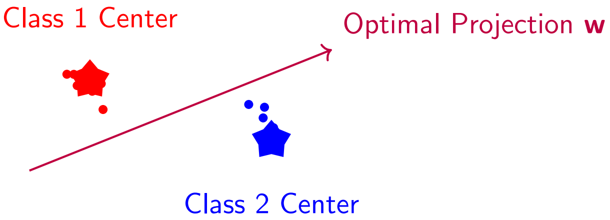
Optimal Projection Strategy

We can project the data to a lower-dimensional subspace by finding the linear combination:

$$Z = \mathbf{w}^T \mathbf{X}$$

that **maximizes between-class variance** while **minimizing within-class variance**.

Key Insight for Dimensionality Reduction II



Key Matrices in LDA

Within-Class Scatter Matrix ($\mathbf{S_W}$)

Measures how spread out points are within each class

$$\mathbf{S_W} = \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$$

Between-Class Scatter Matrix ($\mathbf{S_B}$)

Measures separation between different classes

$$\mathbf{S_B} = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$

The LDA Objective

We want to maximize class separability!

Fisher's Criterion

Find projection vector **w** that maximizes:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S_B} \mathbf{w}}{\mathbf{w}^T \mathbf{S_W} \mathbf{w}}$$

- Numerator: Between-class variance (maximize)
- Denominator: Within-class variance (minimize)
- Solution: Generalized eigenvalue problem

Solving the LDA Problem

Generalized Eigenvalue Problem

$$\mathbf{S_B w} = \lambda \mathbf{S_W w}$$

Solution Steps:

1. Compute $\mathbf{S_W}$ and $\mathbf{S_B}$
2. Solve $\mathbf{S_W}^{-1}\mathbf{S_B w} = \lambda \mathbf{w}$
3. Select eigenvectors with largest eigenvalues
4. Project data: $\mathbf{y} = \mathbf{W}^T\mathbf{x}$

Maximum dimensions: $min(c-1, d)$ where c = number of classes, d = original dimensions

LDA Intuition: Maximizing Separation

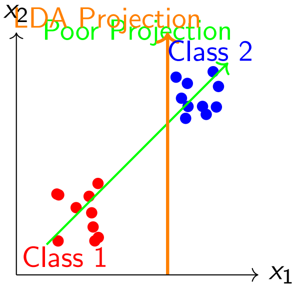


Figure: LDA finds the projection that best separates the classes

2D to 1D Reduction with LDA

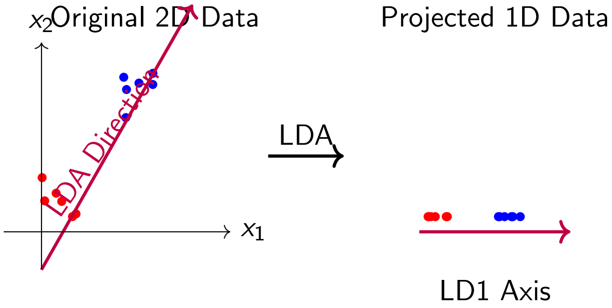


Figure: LDA projects data to 1D while maintaining class separation

LDA Step-by-Step Algorithm

1. **Compute class means**

$$\mu_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

2. **Compute scatter matrices**

$$\mathbf{S_W} = \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$$

$$\mathbf{S_B} = \sum_{i=1}^{c} n_i(\mu_i - \mu)(\mu_i - \mu)^T$$

3. **Solve eigenvalue problem**

$$\mathbf{S_W}^{-1}\mathbf{S_B}\mathbf{w} = \lambda\mathbf{w}$$

4. **Select top k eigenvectors** $(k \leq c - 1)$
5. **Project data**

$$\mathbf{Y} = \mathbf{XW}$$

Limitations and Considerations

Limitations of LDA

- **Linear assumptions**: Assumes linear decision boundaries
- **Normality assumption**: Works best with normally distributed data
- **Equal covariance**: Assumes equal covariance matrices for all classes
- **Maximum dimensions**: Limited to $c - 1$ dimensions
- **Sensitivity to outliers**: Can be affected by extreme values

When to Use LDA

- When you have class labels available
- For classification tasks
- When classes are roughly normally distributed
- When computational efficiency is important

LDA vs PCA

Linear Discriminant Analysis (LDA)	Principal Component Analysis (PCA)
Supervised method	Unsupervised method
Uses class labels	Ignores class labels
Maximizes class separability	Maximizes variance
Better for classification	Better for visualization
Considers between-class and within-class variance	Considers total variance

Kernel PCA: Beyond Linearity I

Motivation and the Core Idea

The Limitation of Standard PCA

- PCA and LDA is a **linear** dimensionality reduction technique.
- It fails to capture complex **non-linear** structures.

The Kernel PCA Intuition

"Bend the space, then do linear PCA."

1. Map data to a **higher-dimensional feature space** $\mathcal{F}$ using a function $\phi(\mathbf{x})$.
2. Perform **standard linear PCA** in this new space $\mathcal{F}$.
3. The non-linear structure in the original space becomes linear in $\mathcal{F}$.

The Kernel Trick

Kernel PCA: Beyond Linearity II

Motivation and the Core Idea

The mapping ϕ is expensive (even infinite).
The **kernel function** $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ lets us compute dot products **without knowing** ϕ!

Kernel PCA in Practice I

The Algorithm and Key Points

The Kernel PCA Algorithm (Simplified)

1. Choose a kernel function k (e.g., RBF, Polynomial).
2. Compute the **Kernel Matrix K** where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.
3. **Center** the kernel matrix in the feature space ($\mathbf{K}_{\text{centered}}$).
4. Solve the eigenvalue problem: $\mathbf{K}_{\text{centered}}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$.
5. The eigenvectors $\boldsymbol{\alpha}_k$ are the principal components in the feature space.
6. Project a new data point $\mathbf{x}_{\text{new}}$ onto the k-th component: $\text{PC}_k(\mathbf{x}_{\text{new}}) = \sum_i \alpha_{k,i} k(\mathbf{x}_i, \mathbf{x}_{\text{new}})$.

Common Kernel Functions

Kernel PCA in Practice II

The Algorithm and Key Points

- **RBF/Gaussian:** $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$
- **Polynomial:** $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y} + c)^d$
- **Sigmoid:** $k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^\top \mathbf{y} + \theta)$

Key Takeaway

Kernel PCA performs **non-linear** dimensionality reduction by applying **linear** PCA in a kernel-induced feature space.

Nonlinear Dimensionality Reduction I

Beyond Linear Subspaces

The Need for Nonlinear Methods

When data lies on a **complex manifold** rather than a linear subspace, nonlinear techniques are essential for preserving intrinsic structure.

Popular Nonlinear Dimensionality Reduction Methods

- **t-SNE (t-Distributed Stochastic Neighbor Embedding)**: Excellent for visualization, preserves local structure
- **UMAP (Uniform Manifold Approximation and Projection)**: Fast, preserves both local and global structure
- **Isomap**: Preserves geodesic distances on the data manifold
- **Autoencoders**: Neural network approach that learns compressed representations

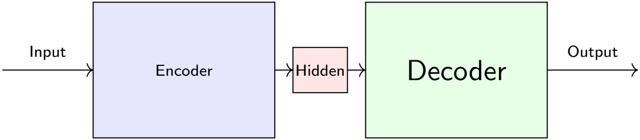
Key Advantage

These methods can **unfold** and **visualize** complex data structures that linear methods like PCA cannot capture effectively.

Autoencoders: The Smart Compression Learning I

Simple Idea

Learn what's truly important by trying to reconstruct the original data



The Learning Process

1. **Squeeze** data down to key features
2. **Remember** what's important
3. **Expand** back to original size
4. **Compare** with original and improve

Perfect For

- **Images**: Learn key shapes and patterns
- **Text**: Capture main ideas and topics
- **Complex data**: Where simple rules don't work
- **Large datasets**: Can learn very detailed patterns

One can use the neural network to learn it.