Kernel PCA

Non-linear extension of PCA using kernel trick

Linear PCA

- Linear transformations only
- Works in original space
- Fast computation
- Limited to linear patterns

Kernel PCA

- Captures non-linear patterns
- Projects to high-dim space
- No explicit computation
- More expressive power

Common Kernel Functions

RBF (Gaussian)
Most popular choice

Polynomial
For polynomial features

Sigmoid
Neural network-like

Key Advantages

✓ Handles non-linear relationships
✓ No explicit feature mapping
✓ Works with complex data structures

Trade-off

More powerful but computationally expensive. Requires careful hyperparameter tuning.

🎯 The Kernel Trick

✨ How It Works

❌ Without Kernel Trick

Explicitly map data to high-dimensional space → Computationally expensive for very high dimensions

✅ With Kernel Trick

Compute inner products in high-dimensional space without explicit mapping → Much more efficient!

💡 Key Insight

Kernel function K(x, y) computes the dot product in feature space without ever computing the coordinates in that space

🔑 Mathematical Beauty

Instead of computing $\varphi(x)\cdot\varphi(y)$ explicitly, we directly compute $K(x, y) = \varphi(x)\cdot\varphi(y)$

🔄 Kernel PCA Process

📋 Step-by-Step Algorithm

1

2

Choose Kernel

Select appropriate kernel function (RBF, polynomial, etc.) and set hyperparameters

Compute Kernel Matrix

Calculate pairwise similarities between all data points using kernel function

3

Center the Matrix

Center the kernel matrix in feature space to ensure zero mean

4

Find Eigenvectors

Compute eigenvalues and eigenvectors of the centered kernel matrix

5

Sort Components

Sort eigenvectors by eigenvalues in descending order

6

Project Data

Project original data onto top k principal components for dimensionality reduction

📊 Visual Example

🎨 Donut-Shaped Data Separation

🔴 Problem: Non-Linearly Separable Data

Imagine red points forming a circle in the center, surrounded by blue points in a donut shape. In 2D, no straight line can separate these classes.

⚙️ Solution: Transform to Higher Dimension

Apply transformation: $f(x, y) = (x, y, 2x^2 + 2y^2)$ to map 2D → 3D

```
f((x, y)) = (x, y, 2x² + 2y²)
```

✨ **Result: Linear Separation**

In 3D space, the classes become linearly separable. Kernel PCA can now find principal components that effectively separate red and blue points!

🌍 Real-World Applications

🚀 Where Kernel PCA Shines

👤 Facial Recognition

Captures complex non-linear facial features for identity verification and security systems

💬 NLP & Text Analysis

Reduces dimensionality of text data for sentiment analysis, document clustering, and classification tasks

🧬 Genomics

Analyzes gene expression data and DNA sequences where non-linear biological relationships exist

📈 Financial Modeling

Captures complex patterns in stock prices and market data for prediction and risk analysis

🖼️ Image Processing

🎤 Speech Recognition

Object detection and recognition by extracting non-linear visual features from image data

Processes audio signals to identify non-linear patterns in speech for better transcription accuracy

🎯 Kernel Selection Guide

🔍 Detailed Kernel Comparison

⭕ RBF (Radial Basis Function) Kernel

Best for: Most general-purpose applications, unknown data patterns

```
K(u, v) = exp(-γ ||u - v||²)
```

Hyperparameter γ: Controls influence radius. Larger γ = more local influence

📈 Polynomial Kernel

Best for: Data with polynomial relationships, specific degree interactions

```
K(u, v) = (γu·v + c)^d
```

Hyperparameters: degree d, coefficient γ, constant c

〰️ Sigmoid Kernel

Best for: Neural network-like transformations

```
K(u, v) = tanh(γu·v + c)
```

Note: Similar to neural network activation functions

💡 Selection Strategy

Use cross-validation to test different kernels and hyperparameters. Start with RBF as default, then experiment with polynomial if domain knowledge suggests polynomial relationships.