Exercise 4: PCA

Objective

To implement Principal Component Analysis (PCA) and Kernel PCA (KPCA) and demonstrate their application on synthetic non-linear datasets.

Introduction

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that projects data onto directions of maximum variance. Kernel PCA (KPCA) extends PCA to capture non-linear relationships by applying kernel functions, such as RBF or polynomial kernels, to project data into higher dimensions before performing PCA.

While PCA works well for linearly separable data, Kernel PCA is more effective in capturing complex non-linear structures, making it particularly useful for datasets like concentric circles or moons.

Algorithm

Steps for PCA:

1. Standardize the dataset.
2. Compute the covariance matrix.
3. Perform eigen decomposition and select top k eigenvectors.
4. Project the data onto the new feature space.

Steps for Kernel PCA:

1. Choose a kernel function (e.g., RBF, polynomial).
2. Compute the kernel (similarity) matrix.
3. Center the kernel matrix.
4. Perform eigen decomposition on the kernel matrix.
5. Select top k eigenvectors and project the data.

Python Program

The following code demonstrates PCA and Kernel PCA on a synthetic non-linear dataset (two moons):

```
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import make_moons
from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import StandardScaler

# Generate synthetic non-linear dataset (two moons)
X, y = make_moons(n_samples=300, noise=0.05, random_state=42)

# Standardize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Apply Kernel PCA with RBF kernel
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=15)
X_kpca = kpca.fit_transform(X_scaled)

# Plot Original Data
plt.subplot(1, 3, 1)
plt.scatter(X[:,0], X[:,1], c=y, cmap='viridis')
plt.title("Original Data")

# Plot PCA
plt.subplot(1, 3, 2)
plt.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap='viridis')
plt.title("PCA Projection")

# Plot Kernel PCA
plt.subplot(1, 3, 3)
plt.scatter(X_kpca[:,0], X_kpca[:,1], c=y, cmap='viridis')
plt.title("Kernel PCA (RBF) Projection")

plt.show()
```

Sample Output

1. **Original Data** – Two interleaved moons.
2. **PCA Projection** – Fails to separate the classes well due to linear assumption.
3. **Kernel PCA (RBF)** – Successfully separates the two moons into linearly separable clusters.

Tasks

Task 1 [CO3] [BTL 2] [2 mark]

Generate very high dimensional feature data. Run the program. Apply PCA and Kernel PCA. Reconstruct the original data and write your inference.

Output : 2025007855-TASK 1

Inference of Task-1:

Principal Component Analysis(PCA): It generally tries to find the lower dimensional surface to project the higher dimensional data.

Kernel PCA : While PCA finds directions of maximum variance assuming linear relationships. Kernel PCA, on the other hand, uses kernel functions (like RBF) to implicitly map data into a higher-dimensional space where linear separation becomes possible—even for non-linear patterns.

Mainly,To reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly or lightly, while retaining the variation present in the dataset, up to the maximum extent.

Firstly,we start with the “make-circles” dataset.

To Increase complexity and simulate a high dimensional real-world scenario,we expand to find it in three ways:

Polynomial Features : Using Polynomial features of degree 3 to capture .

RBF Feature : By using RBF Sampler to project data into a non-linear feature space.

Noise Addition : Gaussian noise (200 dimensions) was introduced to further increase dimensionality and test the robustness of the dimensionality recduction technique.

These transform results in a high dimensional feature space(X_high) with significantly more features than the original data.

Dimensionality reduction and Reconstruction:

PCA(Linearly):

It is a linear technique and assumes that the principal components lie in a linear sub-space.

The PCA projection of the data showed some separation but failed to capture the circular structure of the data.Reconstruction from PCA (inverse transform) retained basic spatial relationships but failed to recover the original circular structure, resulting in linear distortion.

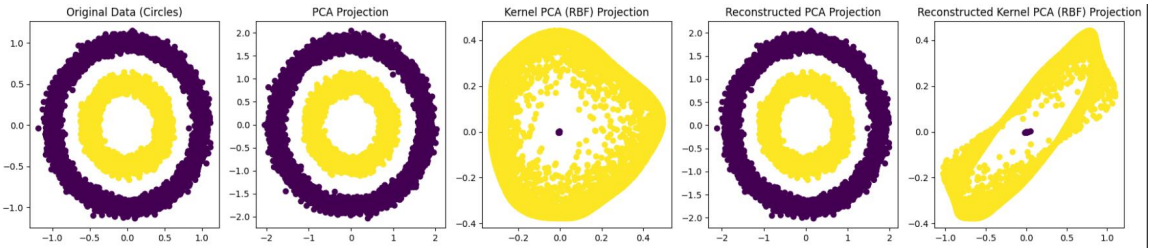
Kernel PCA(Non-Linear with RBF kernel):

Kernel PCA with RBF kernel effectively captured the non-linear structure of the data.

The projection showed a clear unfolding of the concentric circles into a linearly separable structure in the new space.Reconstruction from Kernel PCA (via inverse transform) preserved more of the original circular geometry compared to linear PCA.

Results: Principal component Analysis fails to capture the intrinsic non-linear manifold of the concentric circles. It compresses data based on variance without considering the geometry of class separation.

Kernel PCA with RBF kernel demonstrates superior performance by effectively mapping the data into a space where it can be linearly separated and better reconstructed.



Conclusion:

Kernel PCA significantly out performs Linear PCA for non-linear datasets like concentric circles.

It is capable of capturing and reconstructing non-linear structures that linear PCA cannot.

Visualization and reconstruction results confirm that the RBF kernel enables Kernel PCA to transform and preserve important non-linear relationships in data.

Task 2 **[CO3] [BTL 3] [3 marks]**

Take any simple classifier logistic or SVM. Compare performance with original features and with PCA, kernel PCA reduced features. Take any benchmark dataset. Report your inference.

Output: **2025007855-Task-2**

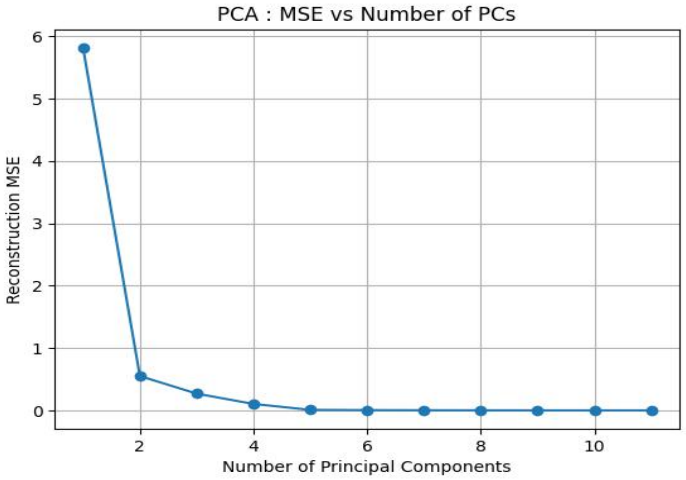
Inference of Task-2:

The main objective of this task is to compare the performance with original features and with PCA, kernel PCA reduced features .so, in order to evaluate their effectiveness we start by using a data set named “winequality-red” . In which there are 11 features which determine the target “wine quality”.

For Principal Component Analysis (PCA) - MSE vs Number of PC's :

When we try to estimate the performance we can see a clear highness in MSE initially in PCA.Gradually on increasing the components we can observe that most data variance is captured in the first 6-8 components.

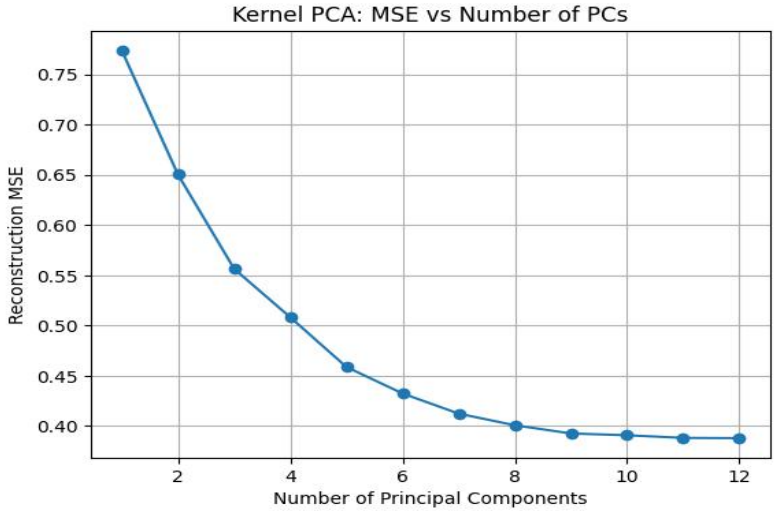
The MSE dropped sharply until 6 components,suggesting diminishing returns beyond this point.



For Kernel PCA with RBF - MSE vs Number of PC's :

Kernel PCA with RBF kernel is applied to standardized data, evaluated similarly from 1 to 11 components .

It is observed that Kernel PCA with RBF can better capture non linear patterns , but the inverse transform is approximate,which increases MSE slightly.



Performance Table : For Time and MSE at different dimensionality levels

Dimensions (%)	PCA Time in sec	PCA MSE	KPCA Time (RBF) in sec	KPCA MSE (RBF)	KPCA Time (Poly) in sec	KPCA MSE (Poly)
10%	**0.0012**	**0.5501**	**0.4418**	**0.6503**	**0.4921**	**0.7545**
20%	**0.0016**	**0.5501**	**0.6398**	**0.6503**	**1.3225**	**0.7545**
40%	**0.0016**	**0.1030**	**0.9183**	**0.5082**	**1.0329**	**0.3865**
60%	**0.0039**	**0.0042**	**0.5727**	**0.4325**	**0.5574**	**0.1912**
80%	**0.0017**	**0.0010**	**0.4910**	**0.4008**	**0.6189**	**0.1451**
100%	**0.0016**	**0.0000**	**0.8465**	**0.3882**	**0.7966**	**0.0851**

We can observe from the table that on reducing the components { 11 PC > 8 PC > 6PC > 4PC > 2PC } there is increase in the MSE { 0.0016 > 0.0017 > 0.0039 > 0.0016 > 0.0012 } .

Conclusion :

Finally, PCA remains a strong for linear data and is optimal when the relationship between features is mostly linear whereas Kernel PCA (RBF) performs better with complex,non-linear data .

PCA remains the most reliable and efficient tool for dimensionality reduction in structured, numerical datasets like wine quality. For most real-world applications,PCA should be the first method to try ,with kernel PCA as an advanced alternative .