Kernel Principle Components Analysis

Yijia Shao,Yida Lyu

November 24, 2024

motivating example

Background

- K-Means
- PCA

Theory

- Kernal
- KPCA

Code

reference

problem

The following dataset is from the website
https://www.kaggle.com/datasets/arnavvvvv/spotify-music

quoting from the website:
'Dataset contains a comprehensive list of the most famous songs and most streamed songs as listed on Spotify.
It provides insights into each song's
Attributes
Popularity
Presence on various music platforms
The dataset includes information such as track name
Artist's name
Release date
Spotify playlists and charts
Streaming statistics
Apple Music presence
Deezer presence
Shazam charts
Various audio features'[3]

We want to category the songs into different clusters with k-means algorithm. However, There are too many dimensions of metrics about the songs in the dataset.To decrease the dimension of the data, we choose the method of KPCA-Kernel Principle components analysis.

K-Means

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.[4]

PCA

Principal components analysis (PCA) is a commonly used dimensionality reduction approach.[2]

Derivation

In PCA, we are given N datapoints $\{x_1, x_2, \cdots, x_N\} \subseteq \mathbb{R}^D$ with D features.
We express x_n as $(x_{n1}, x_{n2}, \cdots, x_{nD})^T$ and $X := [x_1, x_2, \cdots, x_N]^T$,
We need to deal with the problem:

$$\arg \max_{\substack{\{w_i\}_{i=1}^k \subseteq \mathbb{R}^D \text{ is an orthogonal list} \\ \text{of a } k\text{-dimensional subspace of } \mathbb{R}^D}} \sum_{n=1}^{k} \|Xw_n\|^2 \tag{1}$$

Derivation

First, in order to keep the notation uncluttered, we shall assume that we have already subtracted the sample mean from each of the vectors x_n,
s.t.

$$\frac{1}{N}\sum_{n=1}^{N} x_n = 0$$

Derivation

Let's take the first principal component as an example:

$$u_1 := \arg \max_{w_1 \in \mathbb{R}^D, \|w_1\|=1} \|Xw_1\|^2$$

Let

$$S = \frac{1}{N} X^T X = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^T, \tag{2}$$

Then

$$\|Xw_1\|^2 = w_1^T S w_1$$

We know that u_1 is an normalised eigenvector of S,

$$Su_1 = \lambda_1 u_1$$

Derivation

From the class, we know that we can get the answer by Spectral Decomposition:

$$S = Q\Lambda Q^T$$

Then, by SVD,

$$X = U\Sigma V^T,\ V = Q_1$$

The first k columns of V give the answer of PCA.

Kernel

Kernel method is a way to increase dimension.

$$\phi(x) : \mathbb{R}^D \longrightarrow \mathbb{R}^M,\ D < M$$

Besides, we define the kernel function

$$k(x, x') := \phi(x)\phi(x')$$

In the latter analysis, we only need to know information about k instead of ϕ.
We use the Gaussian kernel function for this dataset:

$$k(x, x') = \exp(-\|x - x'\|^2)$$

Certralization

The data matrix is

$$X = (x_1, x_2, \ldots, x_N)^T$$

Then

$$\phi(X) = (\phi(x_1), \phi(x_2), \ldots, \phi(x_N))T$$

As in PCA, we process the high-dimensional dataset obtained from ϕ mapping.
The first step is also centralization. Let

$$\widetilde{\phi}(x_n) = \phi(x_n) - \frac{1}{N}\sum_{n=1}^{N}\phi(x_n) \tag{3}$$

Certralize

Let

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

Certralize

Then

$$\widetilde{K}_{nm} = \widetilde{\phi}(x_n)^T \widetilde{\phi}(x_m) \tag{4}$$

$$= \phi(x_n)^T \phi(x_m) - \frac{1}{N}\sum_{l=1}^{N} \phi(x_n)^T \phi(x_l)$$

$$-\frac{1}{N}\sum_{l=1}^{N} \phi(x_l)^T \phi(x_m) + \frac{1}{N}\sum_{l=1}^{N}\sum_{j=1}^{N} \phi(x_j)^T \phi(x_l) \tag{5}$$

$$= k(x_n, x_m) - \frac{1}{N}\sum_{l=1}^{N} k(x_n, x_l)$$

$$-\frac{1}{N}\sum_{l=1}^{N} k(x_l, x_m) + \frac{1}{N}\sum_{l=1}^{N}\sum_{j=1}^{N} k(x_j, x_l) \tag{6}$$

Kernal

i.e.

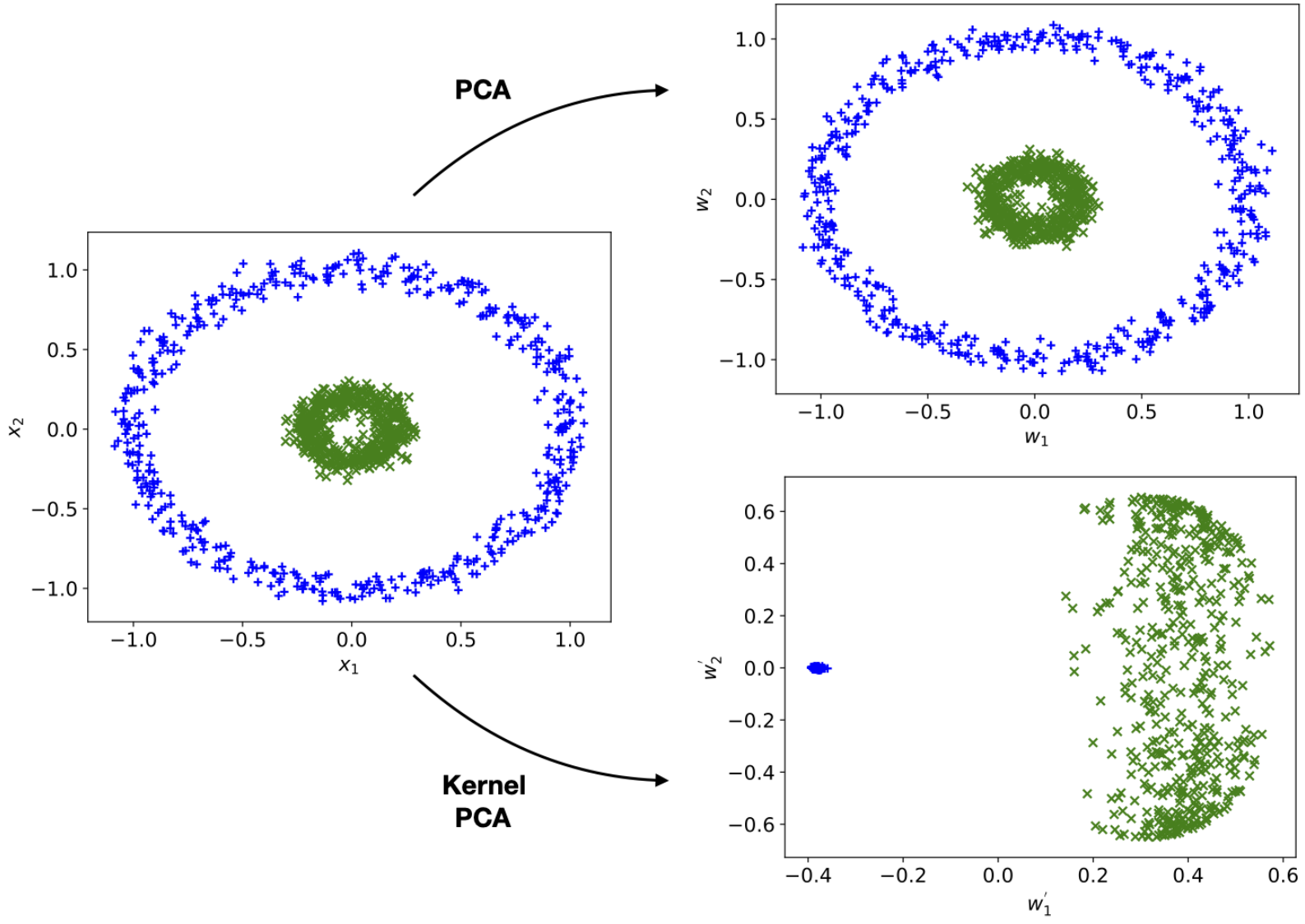
$$\widetilde{K} = K - 1_N K - K 1_N + 1_K 1_N, \quad 1_N = \left(\frac{1}{N}\right)_{N \times N} \tag{7}$$

KPCA

kernel principal component analysis (kernel PCA) is an extension of principal component analysis (PCA) using techniques of kernel methods[4]

KPCA

In some condition, we can not get the principle components directly from the process of PCA, for example:



KPCA

In this case, we can introduce the kernel method to add the dimension first, like $(x_0, x_1) \longrightarrow (x_0, x_1, x_0^2 + x_1^2)$, then we can find out the principle component.

Derivation

$$X = (x_1, x_2, \ldots, x_N)^T$$

$$\phi(X) = (\phi(x_1), \phi(x_2), \ldots, \phi(x_N))^T$$

$$S = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T = \frac{1}{N}\phi(X)^T\phi(X) \in \mathbb{R}^{M \times D}$$

Derivation

We need to deal with

$$Sv_i = \lambda_i v_i, \;\; i = 1, 2, \cdots, M \tag{8}$$

i.e.

$$\frac{1}{N} \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T v_i = \lambda_i v_i \tag{9}$$

Derivation

Let

$$a_{in} = \frac{1}{\lambda_i N} \phi(x_n)^T v_i \tag{10}$$

Then

$$v_i = \sum_{n=1}^{N} a_{in} \phi(x_n) \tag{11}$$

By plugging back, we can get

$$\frac{1}{N} \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \sum_{m=1}^{N} a_{im} \phi(x_m) = \lambda_i \sum_{n=1}^{N} a_{in} \phi(x_n) \tag{12}$$

Derivation

Multiply $\phi(l)$ on the both sides

$$\frac{1}{N}\sum_{n=1}^{N} k(x_l, x_n)\sum_{m=1}^{N} a_{im}k(x_n, x_m) = \lambda_i \sum_{n=1}^{N} a_{in}k(x_l, x_n) \tag{13}$$

i.e.

$$\frac{1}{N}K^2 a_i = \lambda_i K a_i, \text{ where } a_i = (a_{i1}, \cdots, a_{iN})^T \tag{14}$$

The solution of this equation is obtained from the eigenvectors of K with eigenvalue 0 and the following equation:

$$K a_i = \lambda_i N a_i \tag{15}$$

Derivation

Then, we need to normalize the eigenvectors,

$$\begin{aligned} v_i^T v_i &= \sum_{n=1}^{N}\sum_{m=1}^{N} a_{in} a_{im} \phi(x_n)^T \phi(x_m) \\ &= a_i^T K a_i \\ &= \lambda_i N a_i^T a_i \end{aligned} \tag{16}$$

Finally, the projection of a point x onto the principal component v_i is

$$\begin{aligned}
\frac{1}{\sqrt{v_i^T v_i}}\phi(x)^T v_i &= \frac{1}{\sqrt{\lambda_i N a_i^T a_i}}\sum_{n=1}^{N} a_{in}\phi(x)^T\phi(x_n) \\
&= \sum_{n=1}^{N}\frac{a_{in}}{\|a_i\|\sqrt{\lambda_i N}}k(x,x_n)
\end{aligned}$$

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('Popular_Spotify_Songs.csv',
    encoding='latin1')
df.head()
```

Listing 1: Python

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	str
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	14138
1	LALA	Myke Towers	1	2023	3	23	1474	48	13371
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	14000
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	80084
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	30323

5 rows × 24 columns

```python
df.shape
```

Listing 2: Python

```
(953, 24)
```

```python
numeric_cols = df.select_dtypes(['int64','float64']).columns
df = df[numeric_cols]
df.shape
```

Listing 3: Python

```
(953, 17)
```

```python
mean = df.mean()
std = df.std()
outlier_indices = (df < mean - 3 * std) | (df > mean + 3 * std)
df[outlier_indices] = np.nan
df = df.dropna()
df = df.drop_duplicates()
df.shape
```

Listing 4: Python

```
(779, 17)
```

```python
X = df.to_numpy()
N = X.shape[0]
```

Listing 5: Python

```python
def G_kernel_func(x, y):
    return np.exp(-np.linalg.norm(x - y) ** 2)
```

Listing 6: Python

```python
def kernel_matrix(X, kernel_func):
    K = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            K[i, j] = kernel_func(X[i], X[j])
    return K
```

Listing 7: Python

```python
def center_kernel_matrix(K):
    ones_N = np.ones((N, N)) / N
    return K - ones_N @ K - K @ ones_N + ones_N
        @ K @ ones_N
```

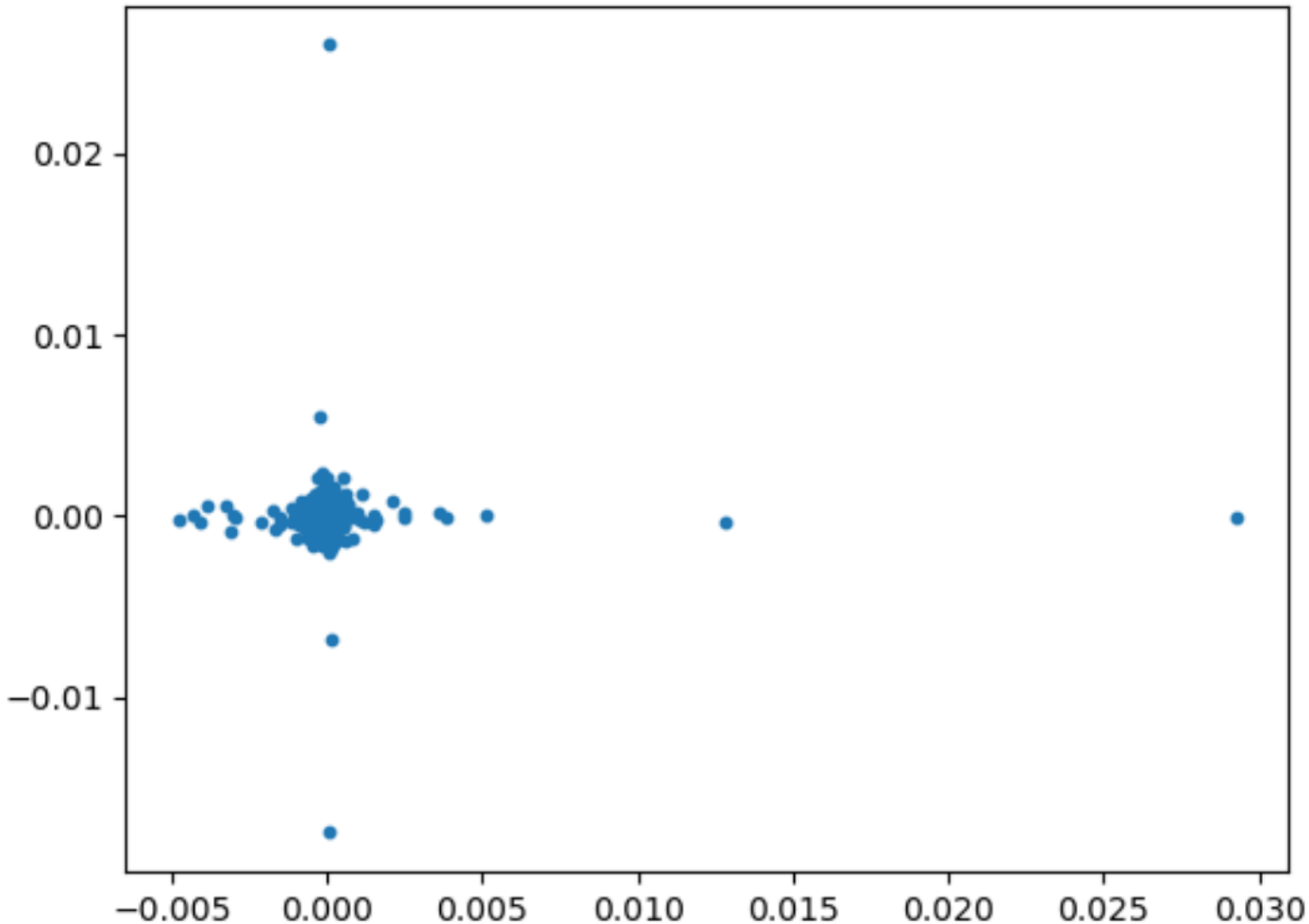
Listing 8: Python

```python
def KPCA(X, n):
    K = kernel_matrix(X, G_kernel_func)
    K = center_kernel_matrix(K)
    L, A = np.linalg.eigh(K)
    for i in range(N):
        if L[i] > 0:
            A[:, i] /= np.sqrt(L[i] * N) * np.
                linalg.norm(A[:, i])
    idx = np.argsort(L)[::-1]
    A = A[:, idx]
    return K @ A[:, :n]
```

Listing 9: Python

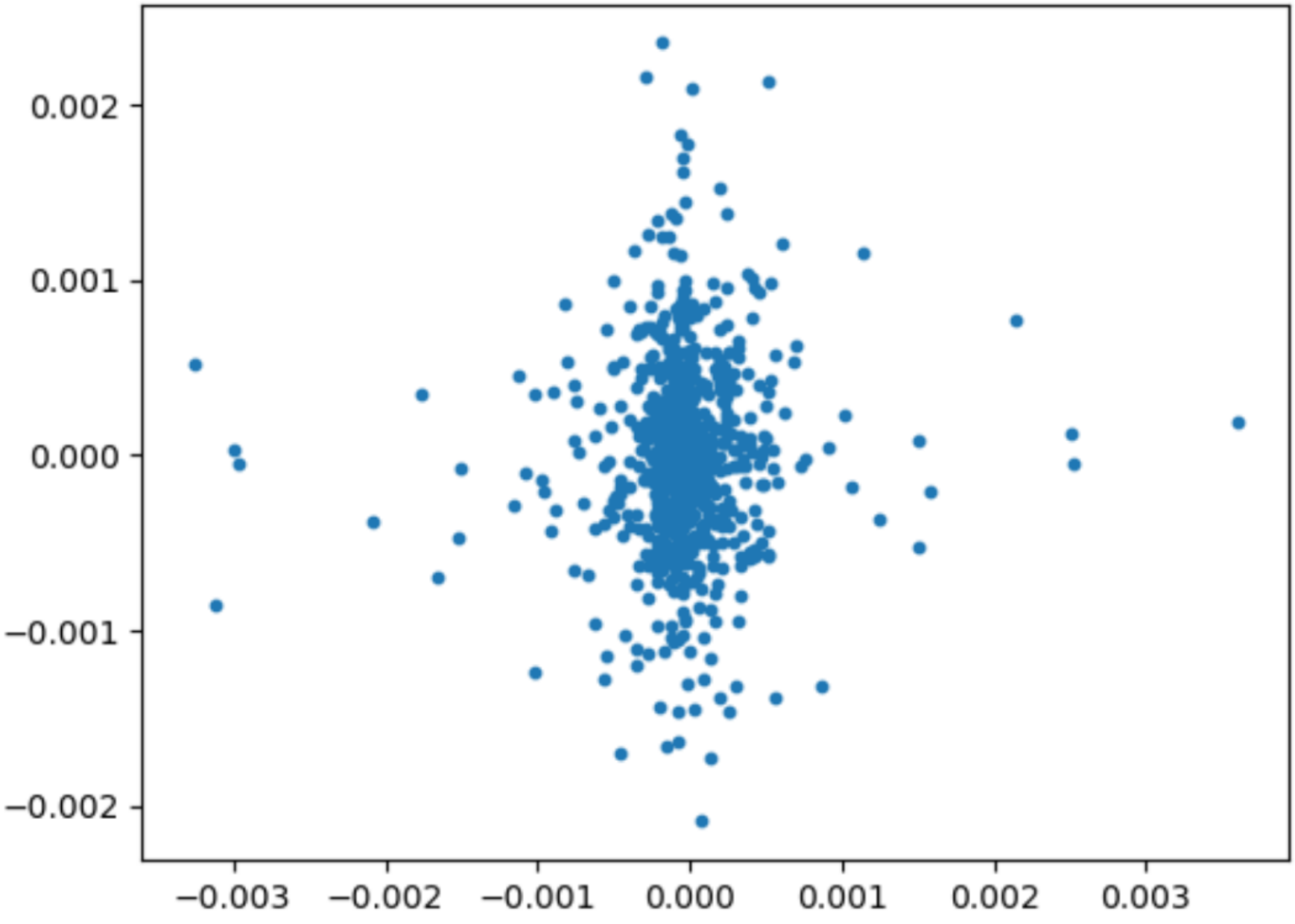
```
1 T = KPCA(X, 2)
2 plt.scatter(T[:,0], T[:,1], s=10)
3 plt.show()
```

Listing 10: Python



```
mean = np.mean(T, axis=0)
std = np.std(T, axis=0)
outlier_indices = (T < mean - 3 * std) | (T >
    mean + 3 * std)
T = T[~np.any(outlier_indices, axis=1)]
plt.scatter(T[:,0], T[:,1], s=10)
plt.show()
```

Listing 11: Python

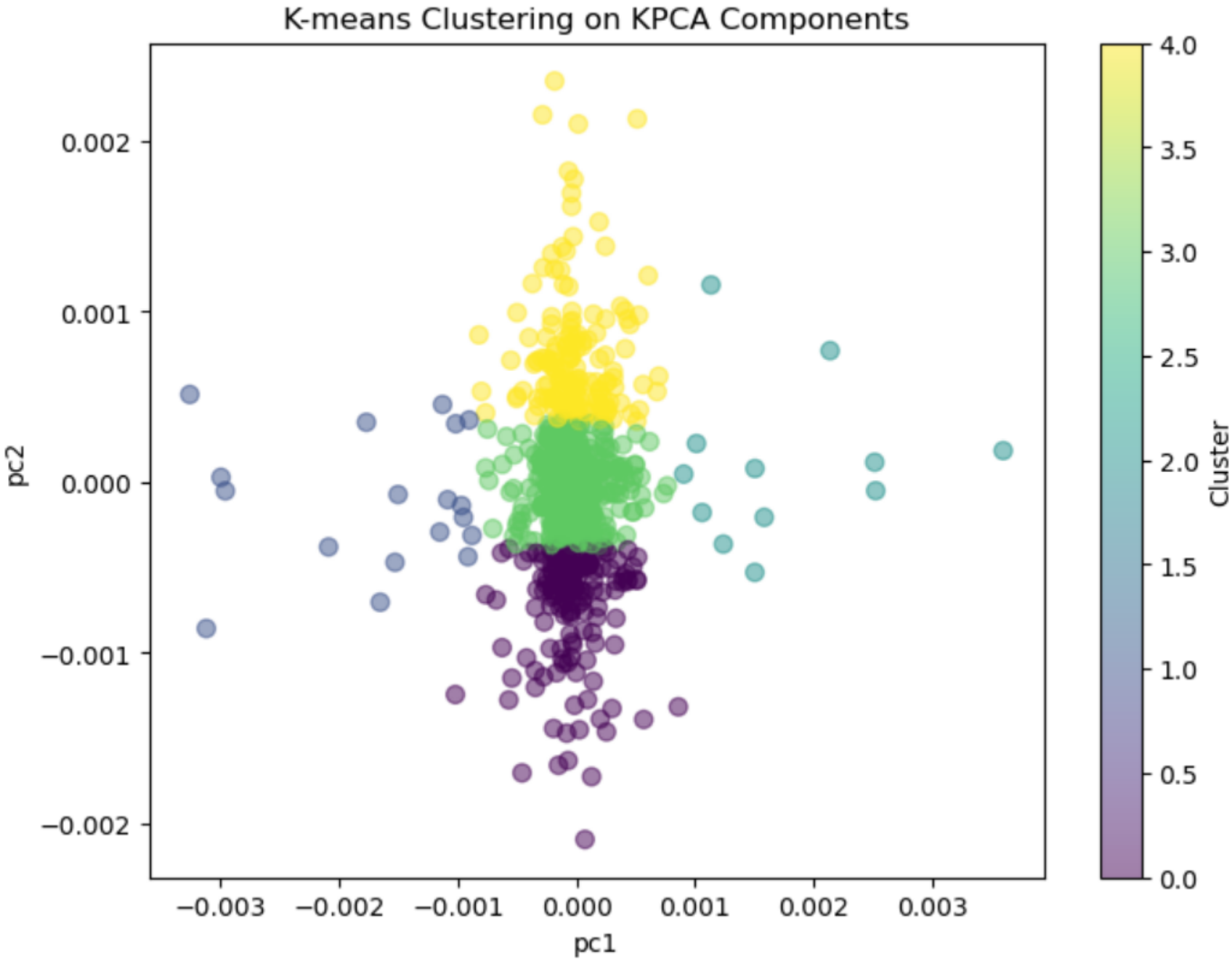

0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.002
−0.001
0.000
0.001
0.002
0.003

```python
from sklearn.cluster import KMeans

kmeans = KMeans(5)
kmeans.fit(T)

plt.figure(figsize=(8, 6))
plt.scatter(T[:, 0], T[:, 1], c=kmeans.labels_,
    cmap='viridis', s=50, alpha=0.5)
plt.xlabel('pc1')
plt.ylabel('pc2')
plt.title('K-means Clustering on KPCA Components
    ')
plt.colorbar(label='Cluster')
plt.show()
```

Listing 12: Python


K-means Clustering on KPCA Components
pc2
pc1
Cluster
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.002
−0.001
0.000
0.001
0.002
0.003
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

- C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006
- **https://mmids-textbook.github.io/index.html**
- https://www.kaggle.com/datasets/arnavvvvv/spotify-music
- https://en.wikipedia.org