Trường Đại học Khoa học tự nhiên – Khoa Công nghệ thông tin.

Đồ án số 01

Trực quan hóa dữ liệu – Data Visualization

Nguyễn Anh Tường
Tháng 10, 2024.

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN THỰC HÀNH CÁ NHÂN SỐ 01

Bộ môn: Trực quan hóa dữ liệu.

Tên đề tài:

"Principle Components Analysis Visualization".

Thuộc nhóm: 16.

Thành viên thực hiện:

22120412 – Nguyễn Anh Tường.

ĐỒ ÁN SỐ 01

Thông tin chung:

1. **Bộ môn:** Trực quan hóa dữ liệu.

2. **Giảng viên lý thuyết:** Thầy Bùi Tiến Lên.

3. **Giảng viên thực hành:** Thầy Lê Nhựt Nam.

4. **Mã lớp:** 22_21.

5. **Số thứ tự nhóm:** 16.

6. **Danh sách thành viên:**

    a. Nguyễn Ngọc Khánh Trân – 22120378.
    b. Trần Đức Trí - 22120387.
    c. Nguyễn Đình Trí – 22120384.
    d. Nguyễn Anh Tường – 22120412.

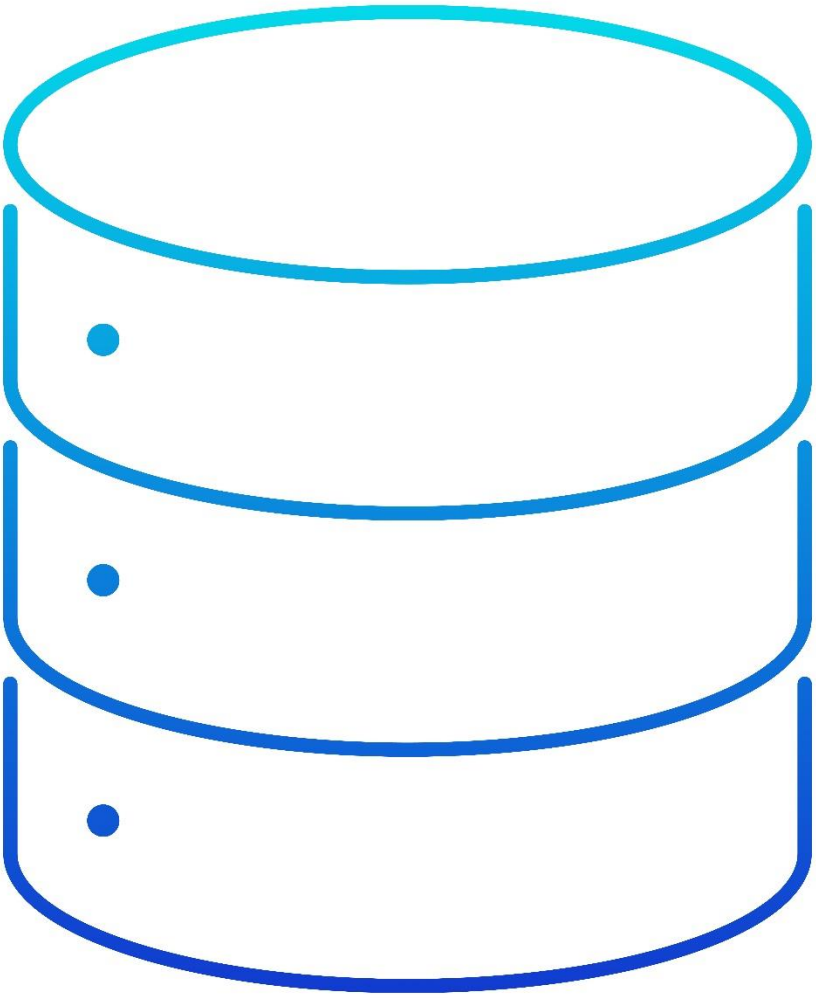
7. **Thành viên thực hiện đề tài:**

    22120412 – Nguyễn Anh Tường.

TABLE OF CONTENT

SECTION 1:

Study about PCA: *động lực, phát biểu vấn đề, thuật toán PCA, bản demo số bằng Python.*



I. Động lực phát triển:

Chúng ta chỉ có thể biểu diễn dữ liệu ở tối đa hai đến ba chiều. Tuy nhiên, trong các vấn đề thực tế thì dữ liệu mà chúng ta thu thập có đến hàng chục, hàng trăm hay thậm chí là hàng ngàn chiều.

Ngoài ra, số lượng các điểm dữ liệu cũng thường rất lớn. Do đó, nếu lưu trữ và làm việc trực tiếp với lượng dữ liệu có chiều cao này sẽ gây ra nhiều khó khăn cả về mặt dung lượng bộ nhớ và tốc độ tính toán, xử lý vấn đề.

Trong những trường hợp trên, chúng ta vẫn có thể áp dụng các thuật toán mà không gặp vấn đề gì, vì các phép toán toán học mà chúng dựa trên hoàn toàn có thể mở rộng cho các không gian có nhiều chiều hơn. Mặt khác, nếu chúng ta muốn thực sự quan sát dữ liệu để thấy các xu hướng trong phân phối hoặc để xem quá trình phân loại hoặc hồi quy diễn ra, điều này trở nên không thể thực hiện được.

Một cách để khắc phục là vẽ dữ liệu theo các cặp chiều, nhưng khi này thì dữ liệu quan sát được sẽ mất tính tổng quát.

Tuy nhiên, trong nhiều tình huống, số chiều thực của dữ liệu thực sự ít hơn nhiều so với số chiều mà dữ liệu được biểu diễn. Hay nói cách khác là ta có thể tưởng tượng tất cả dữ liệu được vẽ trên một tờ giấy duy nhất, vốn chỉ có hai chiều, và sau đó giữ nó theo cách nghiêng trong không gian ba chiều. [1]

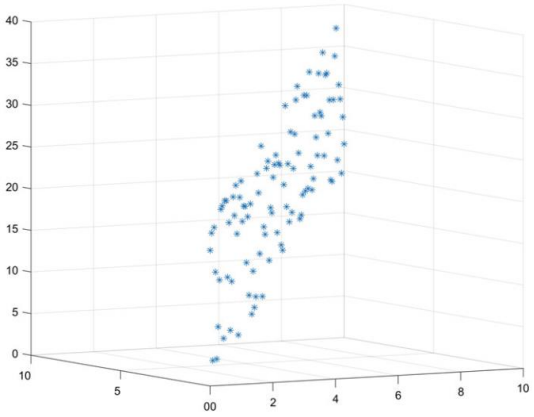


Fig. 3.1 Three-dimensional data containing only two-dimensional information with first perspective

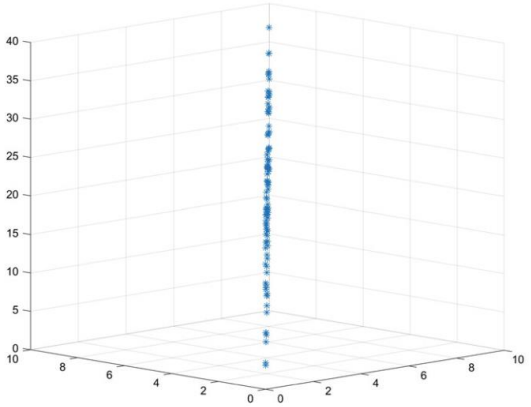


Fig. 3.2 Three-dimensional data containing only two-dimensional information with alternate perspective

Tổng kết lại: PCA được tạo ra với động lực.

1. Giảm đi số chiều của dữ liệu.
2. Giảm đi sự dư thừa thông tin trong dữ liệu.
3. Tăng sự dễ dàng trong việc trực quan hóa và quan sát dữ liệu.
4. Nắm bắt được các thành phần chính, yếu tố quyết định và các mối quan hệ quan trọng dẫn đến kết quả mà không làm thay đổi tính tổng quát, sự biến thiên của dữ liệu.
5. Nén thông tin, đơn giản hóa các quá trình khoa học, quá trình kinh doanh,… phức tạp.

II. Đặt vấn đề:

Vấn đề chính cần giải quyết trong bài toán PCA là làm sao xác định và trích xuất đúng, đủ các chiều quan trọng nhất từ dữ liệu có số chiều cao, nhằm mục đích giảm số lượng chiều trong khi vẫn giữ được phần lớn thông tin (đặc biệt là sự biến thiên) của dữ liệu.

Lưu ý: Thông thường, dữ liệu thực sự có một số thông tin trong tất cả các chiều, nhưng mức độ thông tin trong một số chiều có thể rất nhỏ so với thông tin có mặt trong các chiều khác. Đối với mục đích trực quan hóa và trong các ứng dụng thực tế, việc mất thông tin ở những chiều này là chấp nhận được mà không ảnh hưởng đáng kể đến hiệu suất. [2]

III. Giải thuật PCA:

Kỹ thuật PCA được giới thiệu vào năm 1901 bởi nhà toán học Karl Pearson.

Nguyên lý hoạt động dựa trên điều kiện là khi dữ liệu trong không gian có nhiều chiều hơn được ánh xạ vào dữ liệu trong không gian có ít chiều hơn, thì phương sai của dữ liệu trong không gian có ít chiều hơn phải là lớn nhất. [3]

Phân tích thành phần chính (PCA) là một kỹ thuật giảm chiều xác định một tập hợp các trục trực giao, được gọi là các thành phần chính, nắm bắt được phương sai lớn nhất trong dữ liệu. Các thành phần chính là các tổ hợp tuyến tính của các biến gốc trong tập dữ liệu và được sắp xếp theo thứ tự giảm dần về mức độ quan trọng. Tổng phương sai được nắm bắt bởi tất cả các thành phần chính bằng tổng phương sai trong tập dữ liệu gốc.

Thành phần chính đầu tiên được tính toán sao cho nó giải thích được lượng phương sai lớn nhất trong các đặc điểm ban đầu. Thành phần thứ hai trực giao với thành phần thứ nhất và nó giải thích lượng phương sai lớn nhất còn lại sau thành phần chính thứ nhất. [4]

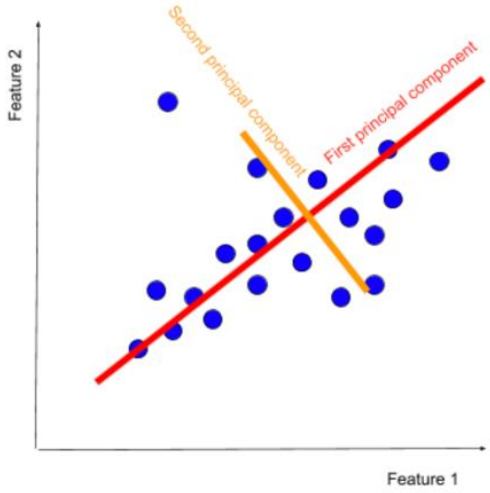


Figure 1. Ảnh mô tả cách xác định các thành phần chính.

Cài đặt giải thuật:
Có nhiều cách để tính PCA: [5]

1. Phân tích riêng của ma trận hiệp phương sai.
2. Phân tích giá trị đơn lẻ của ma trận dữ liệu.
3. Xấp xỉ giá trị riêng thông qua tính toán lặp lại công suất.
4. Tính toán bình phương nhỏ nhất lặp lại phi tuyến tính (NIPALS).
5. Và một số phương pháp khác.

Phương pháp 1 – Phân tích riêng của ma trận hiệp phương sai.

1. **Chuẩn hóa:** chúng ta chuẩn hóa từng đặc trưng để có giá trị trung bình là 0 và phương sai là 1.

$$Z = \frac{X - \mu}{\sigma}$$

Trong đó:

- μ là giá trị trung bình của các đặc trưng độc lập.
- σ là độ lệch chuẩn của các đặc trưng độc lập.

2. **Tính toán ma trận hiệp phương sai:** Ma trận hiệp phương sai là một ma trận vuông, có kích thước *dxd* , trong đó *d* là "chiều" (hoặc đặc trưng hoặc cột, nếu dữ liệu của chúng ta là bảng). Nó hiển thị mối tương quan giữa mỗi đặc trưng.

$$\mathrm{cov}(\mathrm{x}_1, \mathrm{x}_2) = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}}(x_{1i} - \overline{x_1})(x_{2i} - \overline{x_2})}{\mathrm{n} - 1}$$

Giá trị của hiệp phương sai có thể là dương, âm hoặc bằng không.

- Tích cực: Khi x_1 tăng thì x_2 cũng tăng.
- Tiêu cực: Khi x_1 tăng thì x_2 cũng giảm.
- Số không: Không có mối quan hệ trực tiếp.

3. **Tính các giá trị riêng và vector riêng của ma trận hiệp phương sai để xác định thành phần chính.**

4. **Sắp xếp các vectơ riêng từ giá trị riêng cao nhất đến thấp nhất:** Vectơ riêng có giá trị riêng cao nhất là thành phần chính đầu tiên. Thành phần chính với giá trị riêng cao sẽ đóng góp nhiều hơn vào việc mô tả dữ liệu so với các thành phần khác có giá trị riêng thấp hơn.
5. **Chọn số lượng thành phần chính:** Chọn N vectơ riêng hàng đầu (dựa trên các giá trị riêng của chúng) để trở thành N thành phần chính. Số lượng thành phần chính tối ưu vừa mang tính chủ quan vừa phụ thuộc vào vấn đề. Thông thường, chúng ta xem xét lượng tích lũy của phương sai chung được giải thích bởi sự kết hợp của các thành phần chính và chọn số lượng thành phần đó, vẫn giải thích đáng kể phương sai chung. [6][7]

IV. Thử nghiệm với tập dữ liệu số:

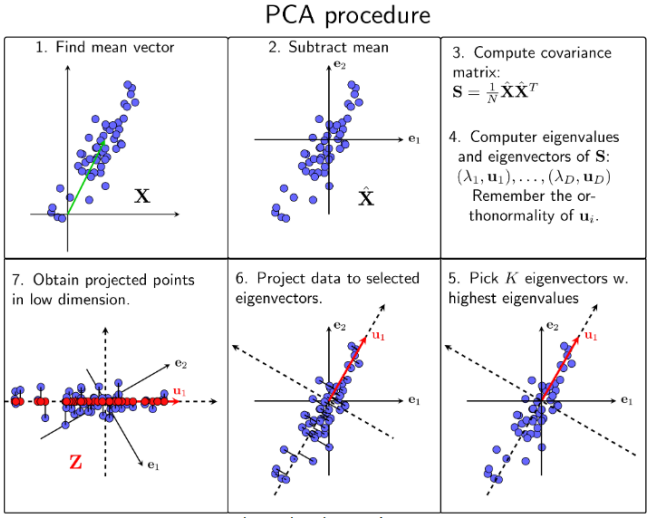


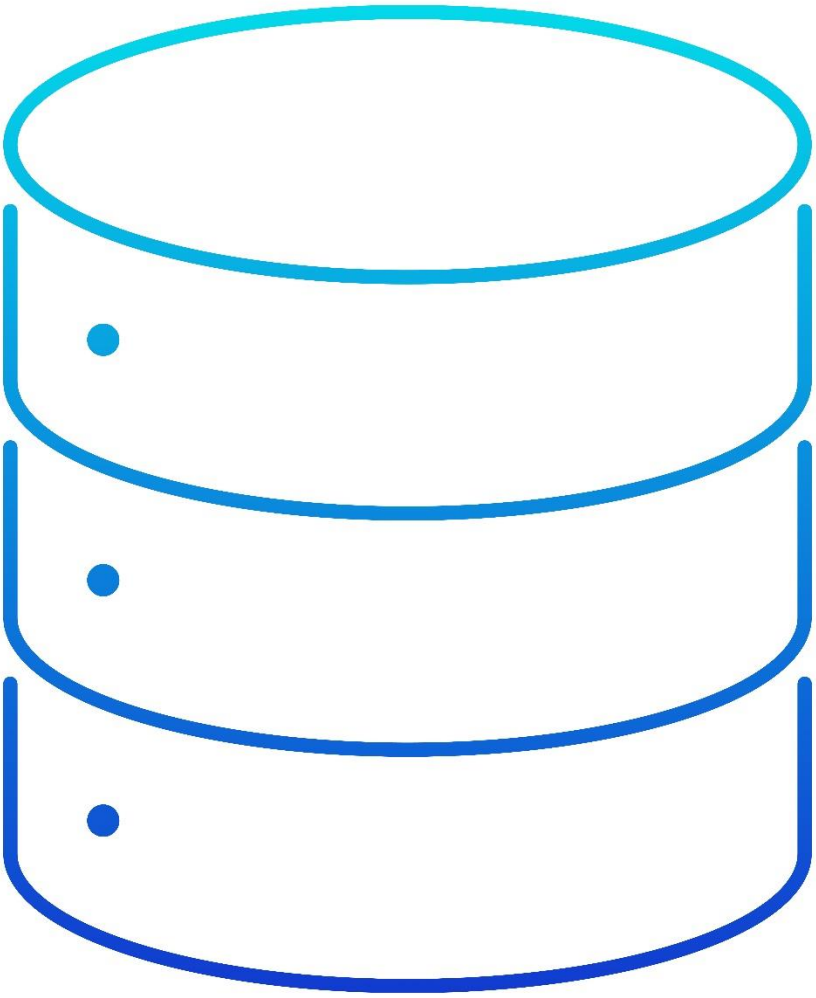
Figure 2. Các bước thực hiện PCA. [8]

```python
# Importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

np.random.seed(0)

data = np.random.rand(10, 15)

columns = ['Feature_1', 'Feature_2', 'Feature_3', 'Feature_4',
'Feature_5','Feature_6', 'Feature_7', 'Feature_8', 'Feature_9',
'Feature_10','Feature_11', 'Feature_12', 'Feature_13', 'Feature_14',
'Feature_15']

df = pd.DataFrame(data, columns=columns)

# Standardize
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply PCA - using Scikit learn
pca = PCA(n_components=5)
pca_data = pca.fit_transform(scaled_data)

pca_df = pd.DataFrame(pca_data, columns=['Principal Component 1', 'Principal
Component 2','Principal Component 3', 'Principal Component 4','Principal
Component 5'])

# Concatenate the original data with the PCA results
original_and_pca_df = pd.concat([df, pca_df], axis=1)

print(original_and_pca_df)

original_and_pca_df.to_csv("output.csv",sep= '\t', header=True)
```

Lưu ý: Trong phần này em sẽ sử dụng thư viện scikit-learn để cài đặt, phần tiếp theo em sẽ làm bằng numpy.

SECTION 2:

Bản chất toán học của PCA.



I. Ý tưởng chính của PCA:

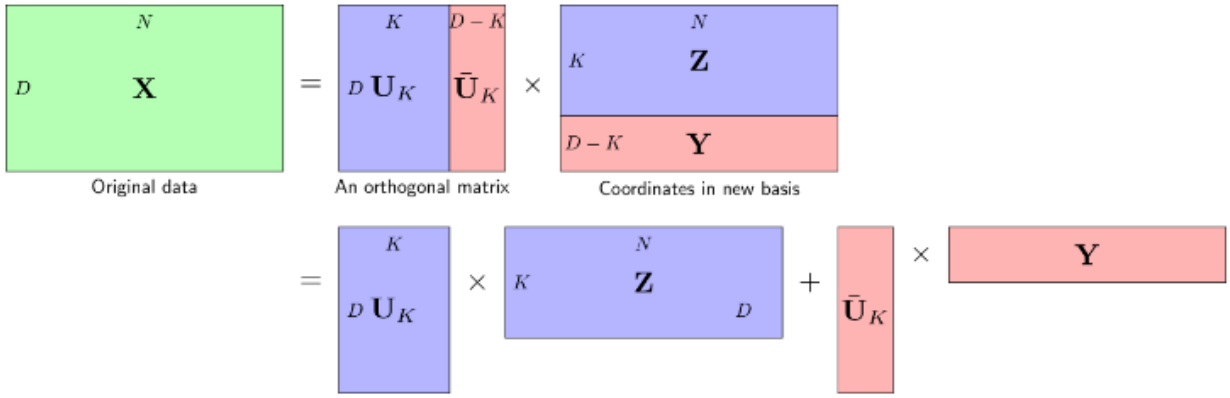


Figure 3. Ý tưởng chính của PCA.

Ý tưởng chính của PCA là tìm một hệ trực chuẩn mới sao cho trong hệ này, các thành phần quan trọng nhất nằm trong K thành phần đầu tiên. [9]

II. Các bước xử lý: [10]

1. Xử lý dữ liệu đầu vào.

- Bắt đầu với tập dữ liệu có *d+1* chiều và bỏ qua nhãn (labels), tạo thành tập dữ liệu có *d* chiều.
- Sau khi bỏ nhãn, chúng ta có một tập dữ liệu với *d* chiều, và đây là tập dữ liệu sẽ được sử dụng để tìm các thành phần chính.

2. Tính trung bình của mỗi chiều.

- Tính giá trị trung bình cho từng chiều của tập dữ liệu. Các cột trong ma trận biểu diễn dữ liệu về từng chiều và mỗi dòng biểu diễn một mẫu.

3. Tính ma trận hiệp phương sai.

- Ma trận hiệp phương sai (covariance matrix) của dữ liệu là phép đo sự phụ thuộc lẫn nhau giữa các chiều dữ liệu.

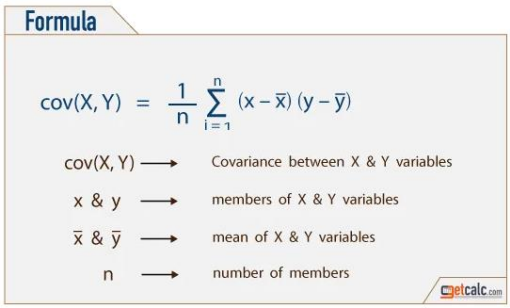


Figure 4. Công thức tính hiệp phương sai giữa hai biến. []

- Phương sai xuất hiện ở đường chéo của ma trận hiệp phương sai. Nếu phương sai cao, điều đó có nghĩa là có sự biến thiên lớn trong dữ liệu ở chiều đó.

4. Tính giá trị riêng (Eigenvalues) và vector riêng (Eigenvectors).

- Giá trị riêng và vector riêng được tính từ ma trận hiệp phương sai (covariance matrix). Vector riêng là hướng trong không gian mà dữ liệu có sự phân tán lớn nhất. Giá trị riêng cho biết mức độ phân tán của dữ liệu theo vector riêng đó.
- Trong đó, các trị riêng của A là nghiệm của phương trình đặc trưng.

$$\det(A-\lambda I) = 0$$

5. Sắp xếp các vectơ riêng theo giá trị riêng giảm dần và chọn k vectơ riêng có giá trị riêng lớn nhất để tạo thành ma trận chiều W *có kích thước d × k* .

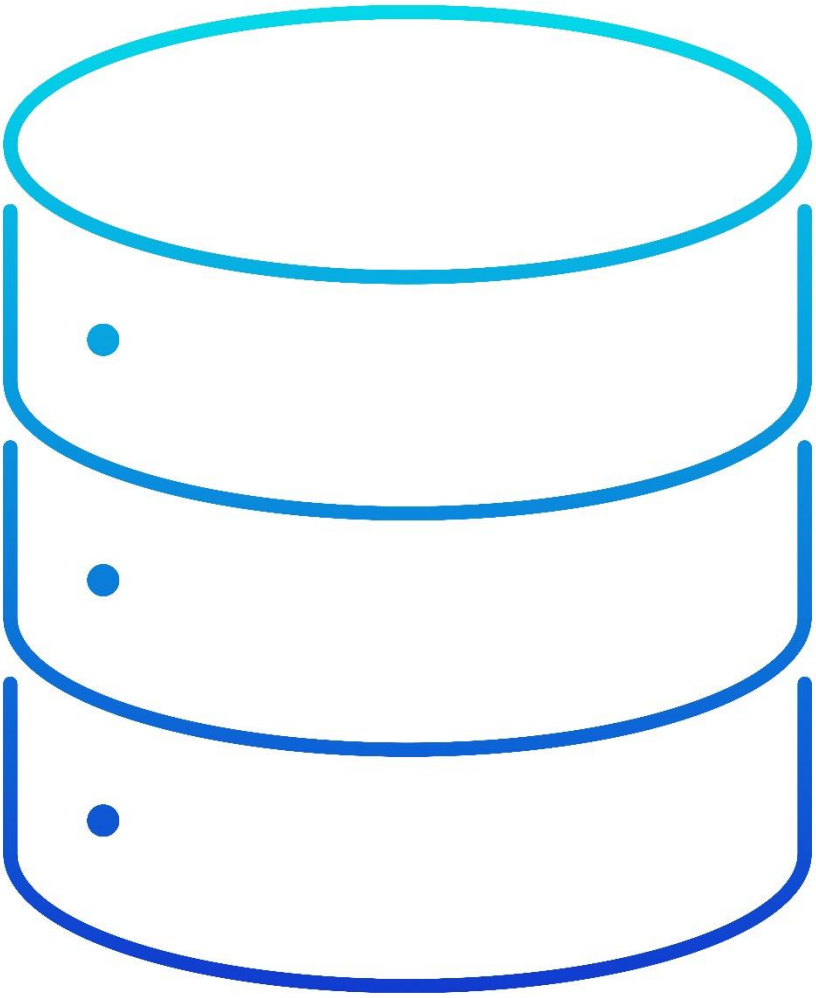
- Sắp xếp các vector riêng theo thứ tự giảm dần của giá trị riêng. Chọn *k* vector riêng tương ứng với *k* giá trị riêng lớn nhất để tạo thành ma trận .
- Những vector riêng có giá trị riêng thấp nhất chứa ít thông tin nhất về phân phối của dữ liệu và có thể được loại bỏ để giảm số chiều.

6. Chuyển đổi dữ liệu sang không gian mới.

- Sử dụng ma trận vector riêng *W* để chuyển đổi dữ liệu sang không gian mới có số chiều thấp hơn thông qua phương trình $y=W'\times x$, trong đó W' là ma trận chuyển vị của *W*.
- Sau khi tính toán, ta sẽ thu được các thành phần chính và biểu diễn dữ liệu trong không gian mới có số chiều nhỏ hơn.

SECTION 3:

Ứng dụng PCA trên tập dữ liệu tự chọn.



I. Giới thiệu tập dữ liệu:

Tên tập dữ liệu: Wine.csv | Gồm 13 đặc trưng để đánh giá rượu cho từng phân khúc khách hàng – với phân khúc khách hàng là đặc trưng thứ 14.

Nguồn: Kaggle.

Kích thước:11.2KB.

Số hàng: 179.

Số cột: 14.

Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenol	Proanthocyanins	Color_Intensity	Hue	OD280	Proline	Customer_Segment
1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	1
2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	1
1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480	1
2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735	1
1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	1

Figure 5. Ảnh mô tả tập dữ liệu.

II. Import các thư viện.

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

III. Đọc dữ liệu.

```
import pandas as pd

def read_wine_csv(file_path):
    dataset = pd.read_csv(file_path, delimiter=',', header=0)

    print(dataset.head())

    return dataset


file_path = 'Wine.csv'

wine_dataset = read_wine_csv(file_path)
```

IV. Gán nhãn và chuẩn hóa các đặc trưng.

```
X = wine_dataset.drop('Customer_Segment', axis=1)  # Data
y = wine_dataset['Customer_Segment']  # Label
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

V. Tiến hành thực hiện PCA bằng thư viện scikit-learn.

➔ Giảm chiều của dữ liệu từ 13 về 2.

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

VI. Trực quan hóa hết quả thông qua thư viện matplotlib.

```
print(X_pca[:50]) # The 50 first rows
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on Wine Dataset')
plt.show()
```

VII. Kết quả.

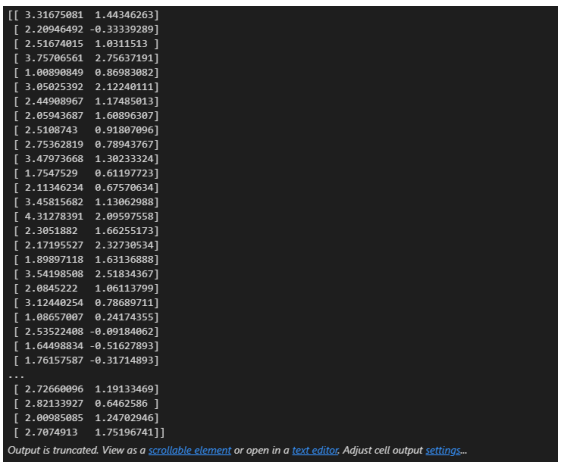


Figure 6. Kết quả dạng bảng của Wine.csv sau khi tiến hành PCA.

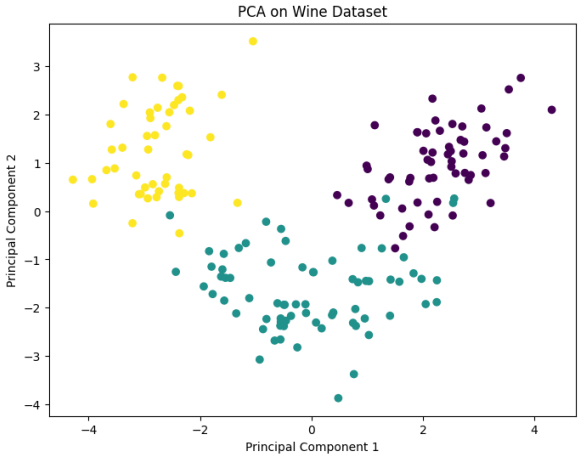


Figure 7. Kết quả dạng biểu đồ của Wine.csv sau khi tiến hành PCA.

VIII. Cài đặt bằng Numpy:

```
# Input processing.
X = wine_dataset.drop('Customer_Segment', axis=1)  # Data.
y = wine_dataset['Customer_Segment']  # label.

#Standardlize the mean.
X_mean = np.sum(X, axis=0) / X.shape[0]
X_mean_standardized = X - X_mean

#Standarlize the variance.
X_std = X_mean_standardized / np.std(X, axis = 0)

#find covariance matrix.
cov_matrix = np.dot(X_std.T, X_std) / ( X_std.shape[0] - 1)

#find eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

#Arrange descending to get the maximum first.
desc_indexing = np.argsort(eigenvalues)[::-1]
desc_eigenvalues = eigenvalues[desc_indexing]
desc_eigenvectors = eigenvectors[:, desc_indexing]

# I got 2 first principle components
needed_eignvectors = desc_eigenvectors[:, : 2]

X_pca = np.dot(X_std, needed_eignvectors)
```

```
#Show 50 first rows
print(X_pca[:50])

#Visualize
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on Wine Dataset')
plt.show()
```

Ở đoạn code trên em làm theo đúng các bước mà em đã trình bày ở trên:

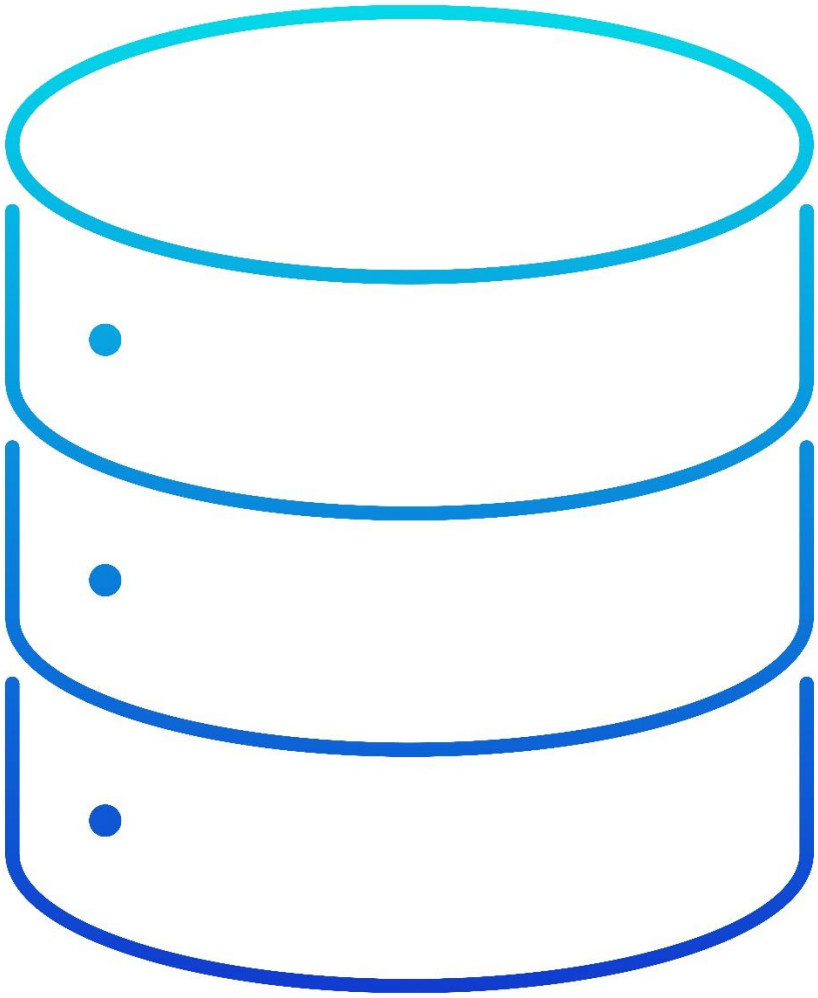
1. Xử lý dữ liệu đầu vào.
2. Tính trung bình của mỗi chiều.
3. Tính ma trận hiệp phương sai.
4. Tính giá trị riêng (Eigenvalues) và vector riêng (Eigenvectors).
5. Sắp xếp các vectơ riêng theo giá trị riêng giảm dần và chọn k vectơ riêng có giá trị riêng lớn nhất để tạo thành ma trận chiều W *có kích thước* $d \times k$.
6. Chuyển đổi dữ liệu sang không gian mới.

Nhận xét:

Kết quả in ra sẽ ngược dấu với việc được tính bằng thư viện nhưng việc này là không có vấn đề vì do hướng của các vector riêng không phải là duy nhất. Hay nói cách khác là ta nhân với 1 hay -1 thì chúng vẫn mô tả chung một hướng. Do các vector riêng chỉ thể hiện hướng chứ không thể hiện chiều cụ thể.

SECTION 4:

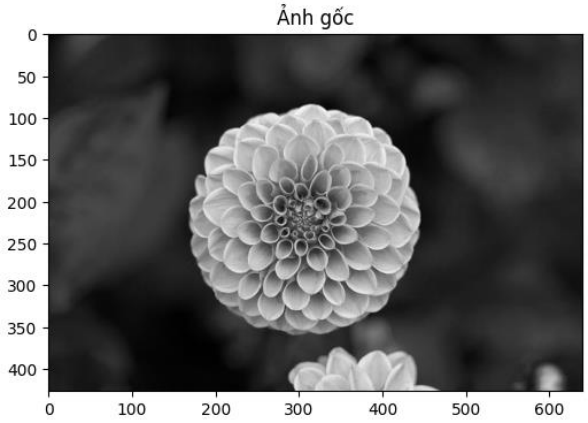
Nghiên cứu ứng dụng của nó trong xử lý dữ liệu trước (ví dụ: xử lý các giá trị bị thiếu)

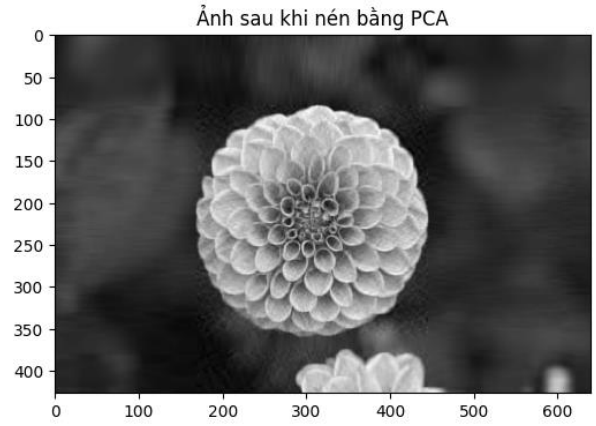


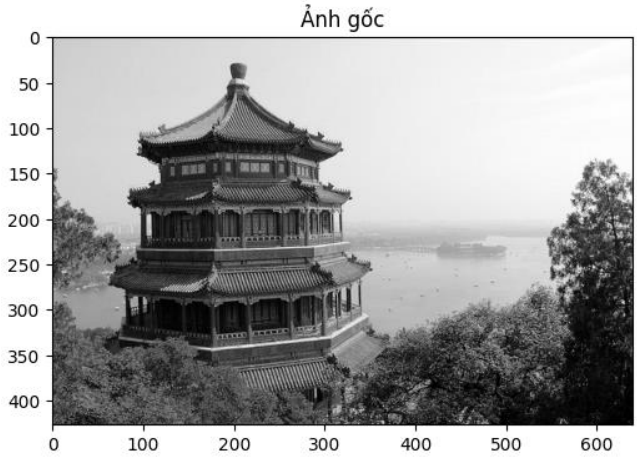
Một số ứng dụng của PCA trong giai đoạn preprocessing.

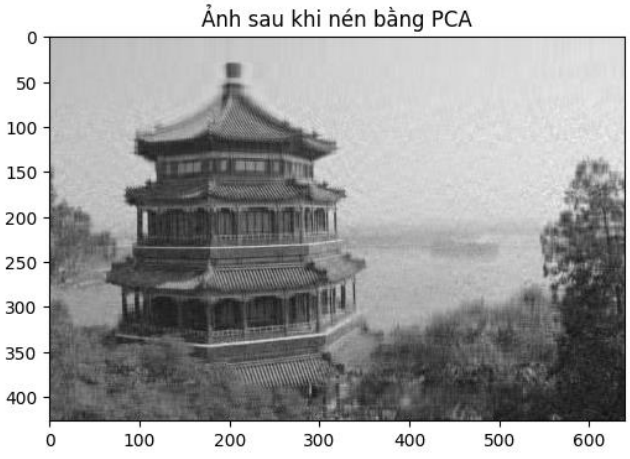
2. Nén hình ảnh: [11]

PCA giúp ta giảm chiều của hình ảnh mà vẫn giữ được các thông tin cần thiết. Từ đó tạo ra các biểu diễn ảnh nhỏ gọn giúp lưu trư và truyền tải dễ dàng hơn.









3. Trực quan hóa dữ liệu đa chiều: [12]

Đối với một tập dữ liệu lớn, với rất nhiều đặc trưng có thể gây khó khăn cho chúng ta ở giai đoạn nhận biết các xu hướng ban đầu, các điểm tập trung dữ liệu,… hay nói đúng hơn là gây khó khăn cho chúng ta trong giai đoạn phân tích ban đầu. Khi này, PCA sẽ giúp chúng ta lọc bỏ thông tin nhiễu, giảm khối lượng dữ liệu cần xử lý và hơn hết là giúp chúng ta trực quan hóa dữ liệu thành những dạng biểu đồ 2D-3D dễ nhìn.

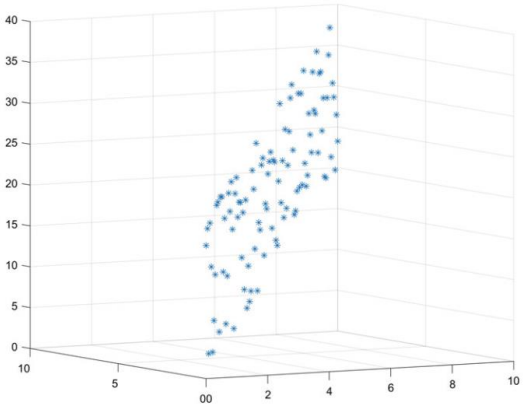


Fig. 3.1 Three-dimensional data containing only two-dimensional information with first perspective

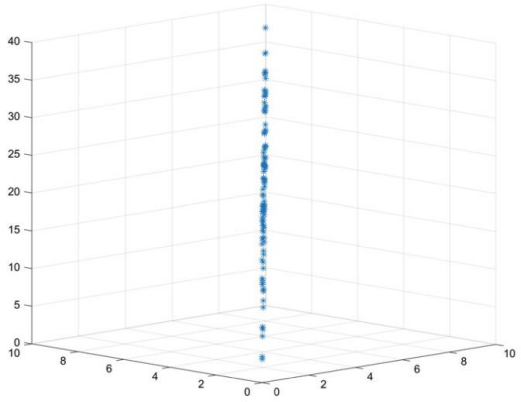


Fig. 3.2 Three-dimensional data containing only two-dimensional information with alternate perspective

4. Lọc tiếng ồn: [13]

PCA có thể loại bỏ nhiễu hoặc thông tin dư thừa khỏi dữ liệu bằng cách tập trung vào các thành phần chính nắm bắt các mẫu cơ bản của đoạn âm thanh. Ví dụ như việc tập trung vào giọng nói và lọc bỏ các tiếng còi xe bên ngoài.

5. Ứng dụng trong công nghệ nhận diện khuôn mặt: [14]

Một mảng các vectơ riêng được sử dụng cho lĩnh vực thị giác máy tính để phát hiện khuôn mặt người được gọi là **eigenface**. PCA là trung tâm của phương pháp eigenface, vì nó tạo ra bộ sưu tập các khuôn mặt có thể có khả năng xảy ra. Phân tích thành phần chính làm giảm độ phức tạp về mặt thống kê của mô tả hình ảnh khuôn mặt trong khi vẫn duy trì các đặc điểm thiết yếu của nó.

6. Trong một số lĩnh vực về tài chính cũng có ứng dụng của PCA.

7. Ứng dụng của PCA trong quá trình tiền xử lý dữ liệu:

a. **Giảm số chiều** trong tập dữ liệu huấn luyện. [15]
b. **Loại bỏ nhiễu** khỏi dữ liệu. Bởi vì PCA được tính toán bằng cách tìm các thành phần giải thích lượng phương sai lớn nhất, nó nắm bắt tín hiệu trong dữ liệu và loại bỏ nhiễu.
c. **Xử lý giá trị thiếu**: PCA có thể được sử dụng để ước lượng và điền vào các giá trị thiếu dựa trên mối quan hệ giữa các đặc trưng khác.