

Phân tích dữ liệu thông minh

Kỹ Thuật giảm chiều

Principal Component Analysis (PCA)

TS. Nguyễn Tiến Huy

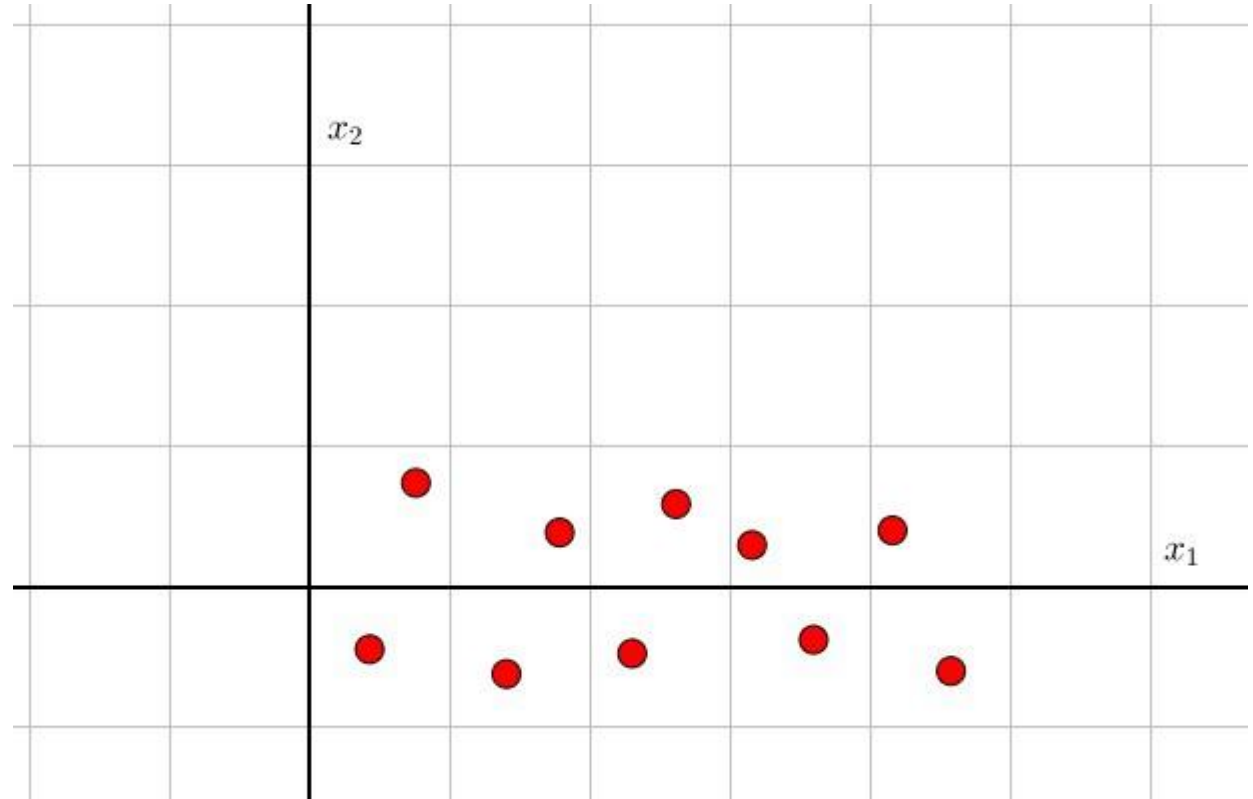
ntienhuy@fit.hcmus.edu.vn

Nội dung

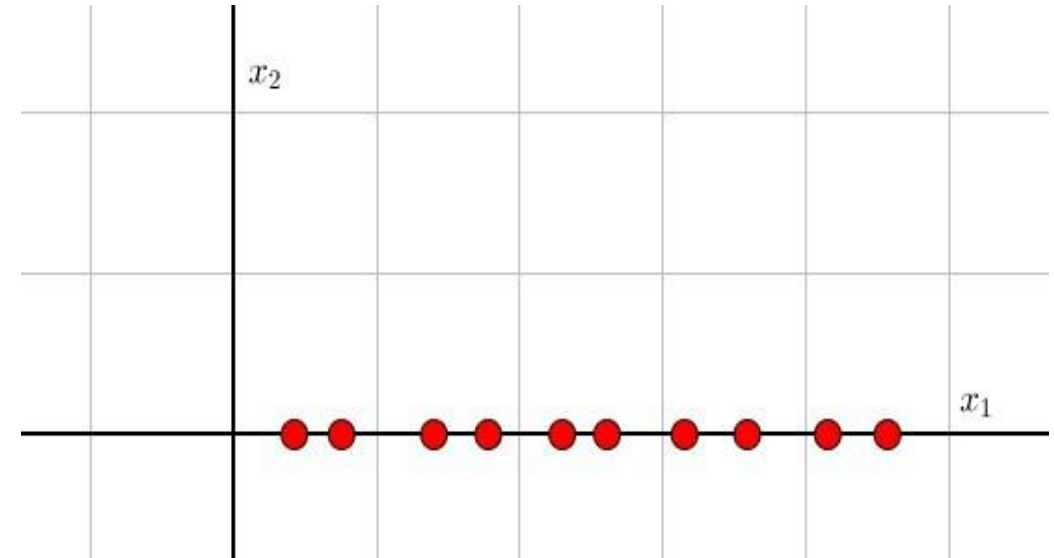
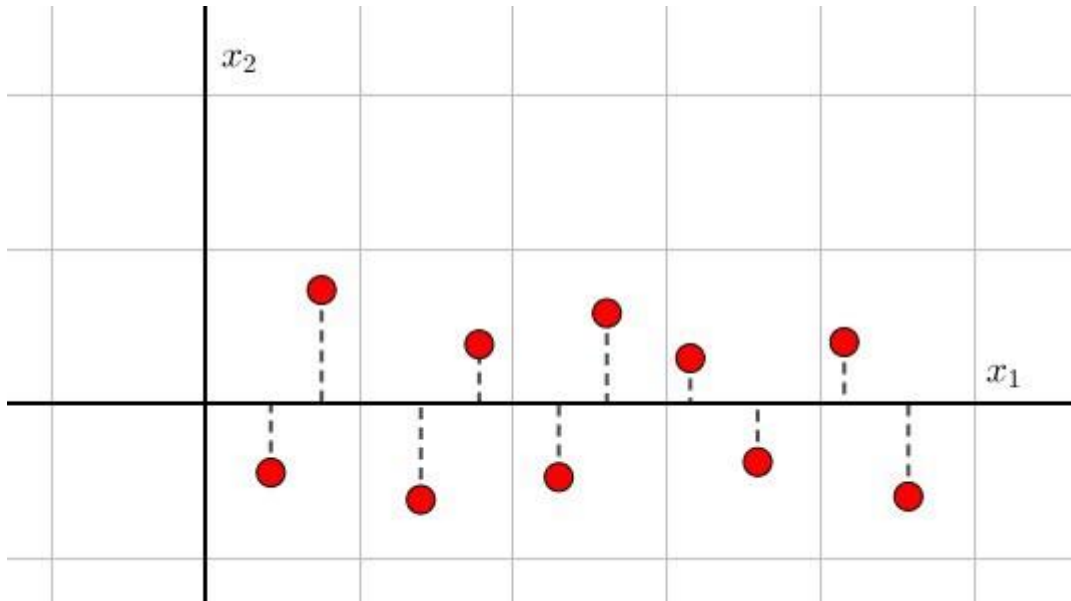
- 1 Giới thiệu PCA
- 2 Ứng dụng trong trực quan hóa dữ liệu (data visualization)

Giới thiệu PCA

- Chuyển dữ liệu n chiều sang dữ liệu m chiều ($m < n$) mà vẫn giữ được nhiều thông tin nhất có thể

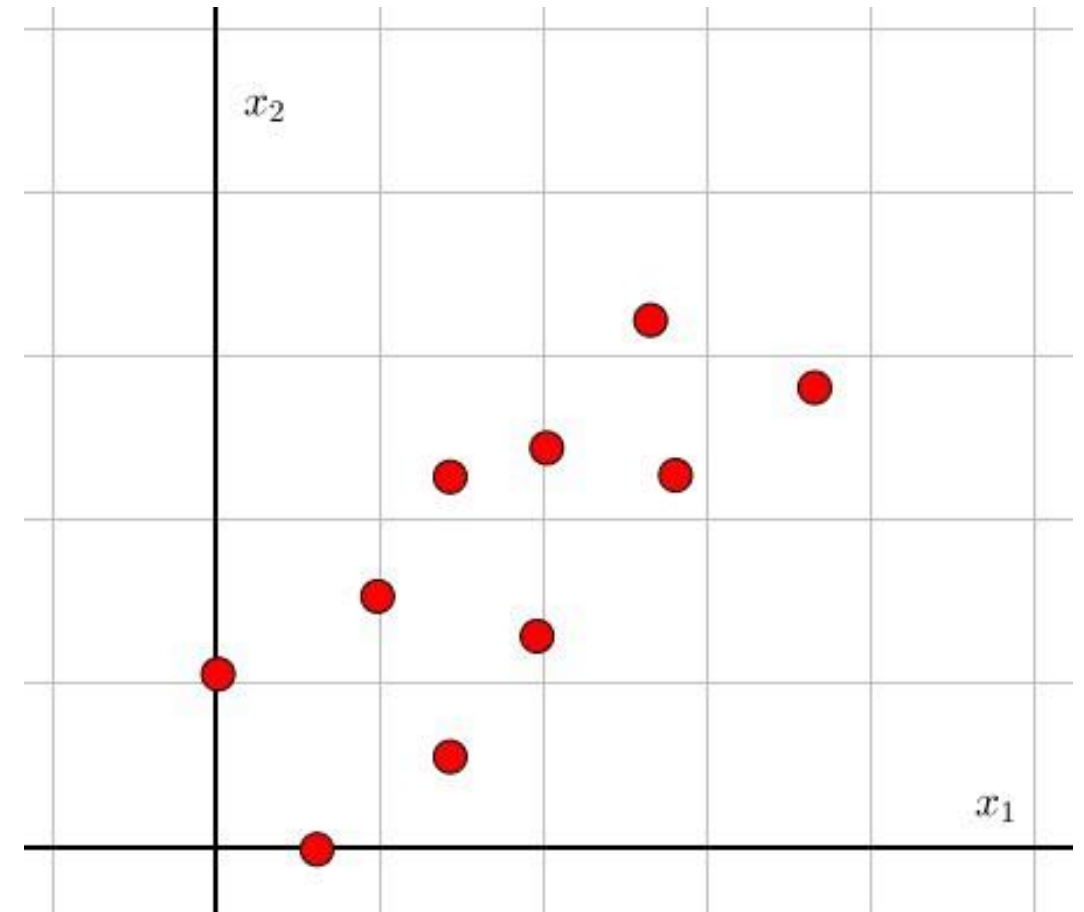


Giới thiệu PCA



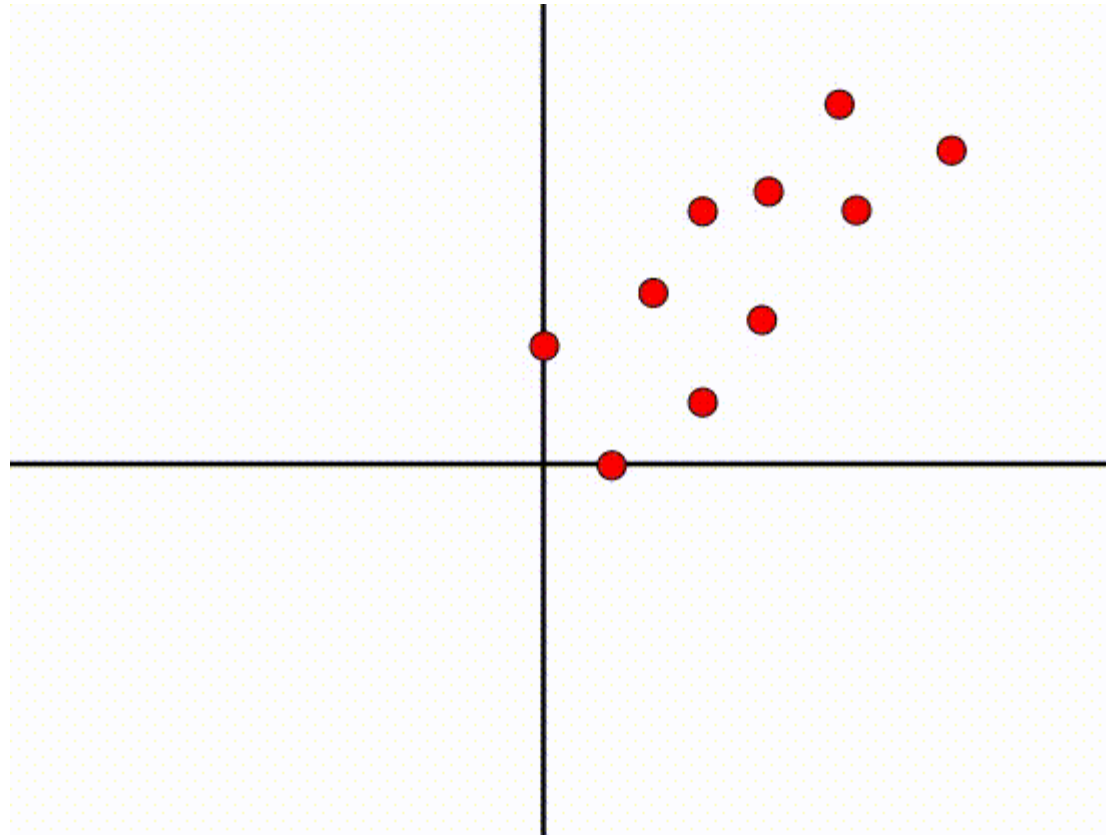
Giới thiệu PCA

- Phải làm gì nếu trục x_1 và x_2 đều không hiệu quả để giảm chiều?



Giới thiệu PCA

- Cần tìm 1 hệ trục mới để việc giảm chiều hiệu quả.



Nền tảng toán của PCA

- **Độ lệch chuẩn** (Standard deviation): khoảng cách trung bình của các điểm dữ liệu đến điểm trung tâm (mean)

$$s = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}}$$

X	$(X - \bar{X})$	$(X - \bar{X})^2$
0	-10	100
8	-2	4
12	2	4
20	10	100
Total		208
Divided by (n-1)		69.333
Square Root		8.3266

Dataset X = [0, 8, 12, 20], s = 8.3266

X_i	$(X_i - \bar{X})$	$(X_i - \bar{X})^2$
8	-2	4
9	-1	1
11	1	1
12	2	4
Total		10
Divided by (n-1)		3.333
Square Root		1.8257

Dataset X = [8, 9, 11, 12], s = 1.8257

Nền tảng toán của PCA

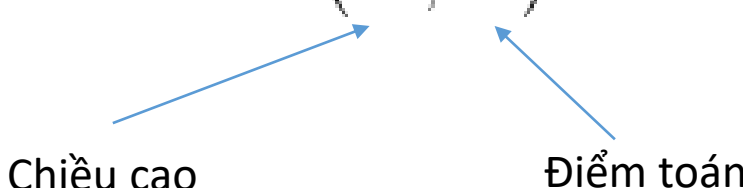
- **Phương sai** (variance) là một cách khác để đo độ lệch dữ liệu.

$$s^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}$$

- Bài tập: Tìm trung bình (mean), độ lệch chuẩn và phương sai cho tập dữ liệu sau:
 - [12, 23, 34, 44, 59, 70, 98]

Nền tảng toán của PCA

- **Hiệp phương sai** (covariance): xem xét sự thay đổi của những chiều dữ liệu với nhau.
- Ví dụ: ta có dữ liệu về chiều cao, điểm toán của các học sinh trong 1 lớp. Ta cần phân tích liệu chiều cao có tác động nào đến điểm không?

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$


Chiều cao

Điểm toán

Nền tảng toán của PCA

- Nếu $\text{cov}(X,Y)$ là số dương \Rightarrow Khi chiều cao tăng, thì điểm toán tăng.
- Nếu $\text{cov}(X,Y)$ là số âm \Rightarrow Khi chiều cao tăng, thì điểm số giảm.
- Nếu $\text{cov}(X,Y)$ bằng 0 \Rightarrow chiều cao và điểm số là độc lập (không ảnh hưởng nhau).

Nền tảng toán của PCA

- **Ma trận hiệp phương sai** (covariance matrix)
- Nếu dữ liệu 3 chiều (x, y, z), ta cần tính ma trận hiệp phương sai như sau:

$$C = \begin{pmatrix} cov(x, x) & cov(x, y) & cov(x, z) \\ cov(y, x) & cov(y, y) & cov(y, z) \\ cov(z, x) & cov(z, y) & cov(z, z) \end{pmatrix}$$

Nền tảng toán của PCA

- Bài tập 2: Ma trận hiệp phương sai
 - A) Cho 5 điểm dữ liệu 2 chiều (x, y). Tính giá trị hiệp phương sai và nhận xét về kết quả này.

Item Number:	1	2	3	4	5
x	10	39	19	23	28
y	43	13	32	21	20

- B) Tính ma trận hiệp phương sai cho dữ liệu 3 chiều sau

Item Number:	1	2	3
x	1	-1	4
y	2	1	3
z	1	3	-1

Nền tảng toán của PCA

- **Vector riêng** (Eigenvectors): vector riêng của 1 ma trận chỉ thay đổi độ dài mà không thay đổi hướng khi nhân với ma trận đó.

The diagram illustrates the concept of eigenvectors and eigenvalues using two matrix multiplication examples. The first example shows a non-eigenvector being multiplied by a matrix, resulting in a vector that is not a scalar multiple of the original. The second example shows an eigenvector being multiplied by the same matrix, resulting in a vector that is a scalar multiple of the original, with the scalar being the eigenvalue. Blue arrows point from the labels to the corresponding parts of the equations.

Non-Eigenvector

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

Eigenvector

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Eigenvalue

Nền tảng toán của PCA

• Vector riêng



```
import numpy as np

A = np.array([[2,3],[2,1]])
w,v = np.linalg.eig(A)

print ("Eigen values",w)
print ("Eigen vectors",v)

print (3/np.sqrt(13)," là giá trị đã chuẩn hóa của 3") #Chuẩn hóa là chia cho chiều dài của vector đó.
print (2/np.sqrt(13)," là giá trị đã chuẩn hóa của 2") #Chuẩn hóa là chia cho chiều dài của vector đó.
```

```
↳ Eigen values [ 4. -1.]
Eigen vectors [[ 0.83205029 -0.70710678]
 [ 0.5547002  0.70710678]]
0.8320502943378437  là giá trị đã chuẩn hóa của 3
0.5547001962252291  là giá trị đã chuẩn hóa của 2
```

Các bước tính PCA

- Bước 1: Chuẩn hóa (trừ trung bình)

		x	y
Data =	2.5	2.4	
	0.5	0.7	
	2.2	2.9	
	1.9	2.2	
	3.1	3.0	
	2.3	2.7	
	2	1.6	
	1	1.1	
	1.5	1.6	
	1.1	0.9	
		x	y
DataAdjust =		.69	.49
		-1.31	-1.21
		.39	.99
		.09	.29
		1.29	1.09
		.49	.79
		.19	-.31
		-.81	-.81
		-.31	-.31
		-.71	-1.01

Các bước tính PCA

- Bước 2: Ma trận hiệp phương sai

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

Các bước tính PCA

- Bước 3: Tính vector riêng và giá trị riêng cho ma trận hiệp phương sai

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Các bước tính PCA

- Bước 4: Biến đổi dữ liệu ma trận riêng

$$\text{Dữ liệu mới} = \text{Ma Trận Riêng} \times \text{Dữ Liệu Đã Chuẩn Hóa}$$

Lưu ý: tùy vào số lượng chiều muốn giảm mà ta sẽ lấy bao nhiêu cột tương ứng của ma trận riêng.

Các bước tính PCA – Tự implement

```
import numpy as np

data = np.array([[2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2, 1, 1.5, 1.1],
                 [2.4, 0.7, 2.9, 2.2, 3, 2.7, 1.6, 1.1, 1.6, 0.9]])

meanX, meanY = (np.mean(data,axis=1)) #Tính trung bình cho từng chiều
dataAdjust = np.array([data[0,:]- meanX,data[1,:]- meanY]) #Chuẩn hóa dữ liệu
cov = np.cov(dataAdjust,ddof=1) #Tính ma trận hiệp phương sai
eigenvalues, eigenvectors = np.linalg.eig(cov) #Tính giá trị riêng và vector riêng
print("Giá trị riêng",eigenvalues)
#Do muốn giảm xuống còn 1 chiều, nên ta chỉ chọn 1 cột trong eigenvectors, ở đây chọn cột 2
selectedEigenvectors = eigenvectors[:,1]
#Chuyển dữ liệu qua hệ trục mới
finaldata = np.dot(selectedEigenvectors,dataAdjust)
print (finaldata)
```

```
➞ Giá trị riêng [0.0490834  1.28402771]
[-0.82797019  1.77758033 -0.99219749 -0.27421042 -1.67580142 -0.9129491
 0.09910944  1.14457216  0.43804614  1.22382056]
```

Các bước tính PCA – Dựa vào thư viện sklearn

```
#sklearn
import sklearn
from sklearn.decomposition import PCA
data = np.array([[2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2, 1, 1.5, 1.1],
                 [2.4, 0.7, 2.9, 2.2, 3, 2.7, 1.6, 1.1, 1.6, 0.9]])
data = np.transpose(data)
pca = PCA(n_components=1)
finaldata = pca.fit_transform(data)

print (finaldata)
```

```
[>] [[-0.82797019]
      [ 1.77758033]
      [-0.99219749]
      [-0.27421042]
      [-1.67580142]
      [-0.9129491 ]
      [ 0.09910944]
      [ 1.14457216]
      [ 0.43804614]
      [ 1.22382056]]
```

Trực quan hóa dữ liệu

- Để biểu diễn dữ liệu nhiều hơn 3 chiều.
 - Ví dụ: dữ liệu về phân loại hoa Iris có 4 chiều ('sepal length','sepal width','petal length','petal width')
- => Giảm về 2 chiều để dễ biểu diễn

Trực quan hóa dữ liệu

- Step 1: Load dữ liệu

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-  
databases/iris/iris.data"

# load dataset into Pandas DataFrame
df = pd.read_csv(url, names=['sepal length', 'sepal width', 'petal  
length', 'petal width', 'target'])
```

	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Original Pandas df (features + target)

Trực quan hóa dữ liệu

- Step 2: Chuẩn hóa dữ liệu

```
from sklearn.preprocessing import StandardScaler

features = ['sepal length', 'sepal width', 'petal length', 'petal width']

# Separating out the features
x = df.loc[:, features].values

# Separating out the target
y = df.loc[:, ['target']].values

# Standardizing the features
x = StandardScaler().fit_transform(x)
```

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Standardization



	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

The array x (visualized by a pandas dataframe) before and after standardization

Trực quan hóa dữ liệu

- Step 3: Giảm dữ liệu còn 2 chiều bằng PCA

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

principalComponents = pca.fit_transform(x)

principalDf = pd.DataFrame(data = principalComponents
                           , columns = ['principal component 1', 'principal
                           component 2'])
```

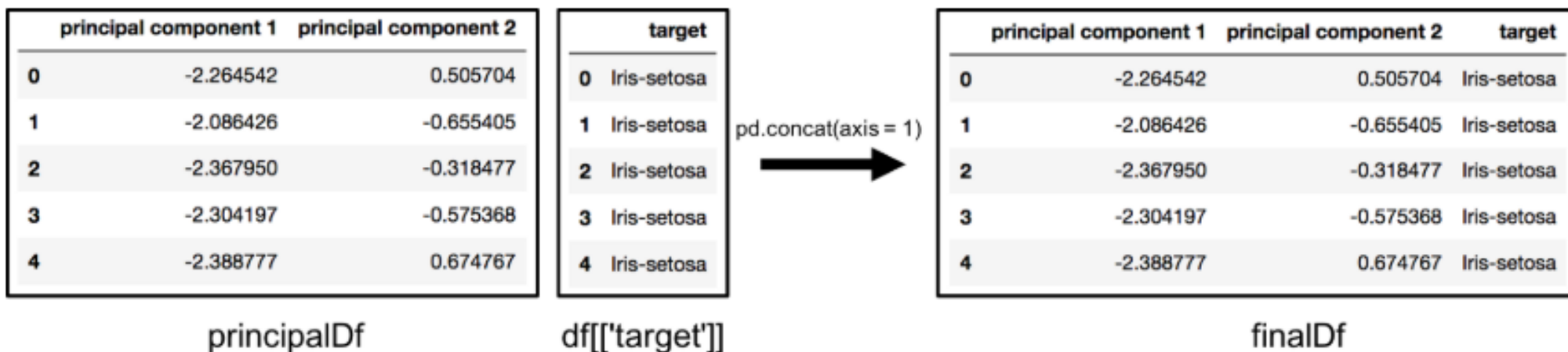
	sepal length	sepal width	petal length	petal width		principal component 1	principal component 2
0	-0.900681	1.032057	-1.341272	-1.312977	0	-2.264542	0.505704
1	-1.143017	-0.124958	-1.341272	-1.312977	1	-2.086426	-0.655405
2	-1.385353	0.337848	-1.398138	-1.312977	2	-2.367950	-0.318477
3	-1.506521	0.106445	-1.284407	-1.312977	3	-2.304197	-0.575368
4	-1.021849	1.263460	-1.341272	-1.312977	4	-2.388777	0.674767

PCA and Keeping the Top 2 Principal Components

Trực quan hóa dữ liệu

- Step 4: Ghép label (nhãn loại hoa) vào dữ liệu

```
finalDf = pd.concat([principalDf, df[['target']], axis = 1)
```



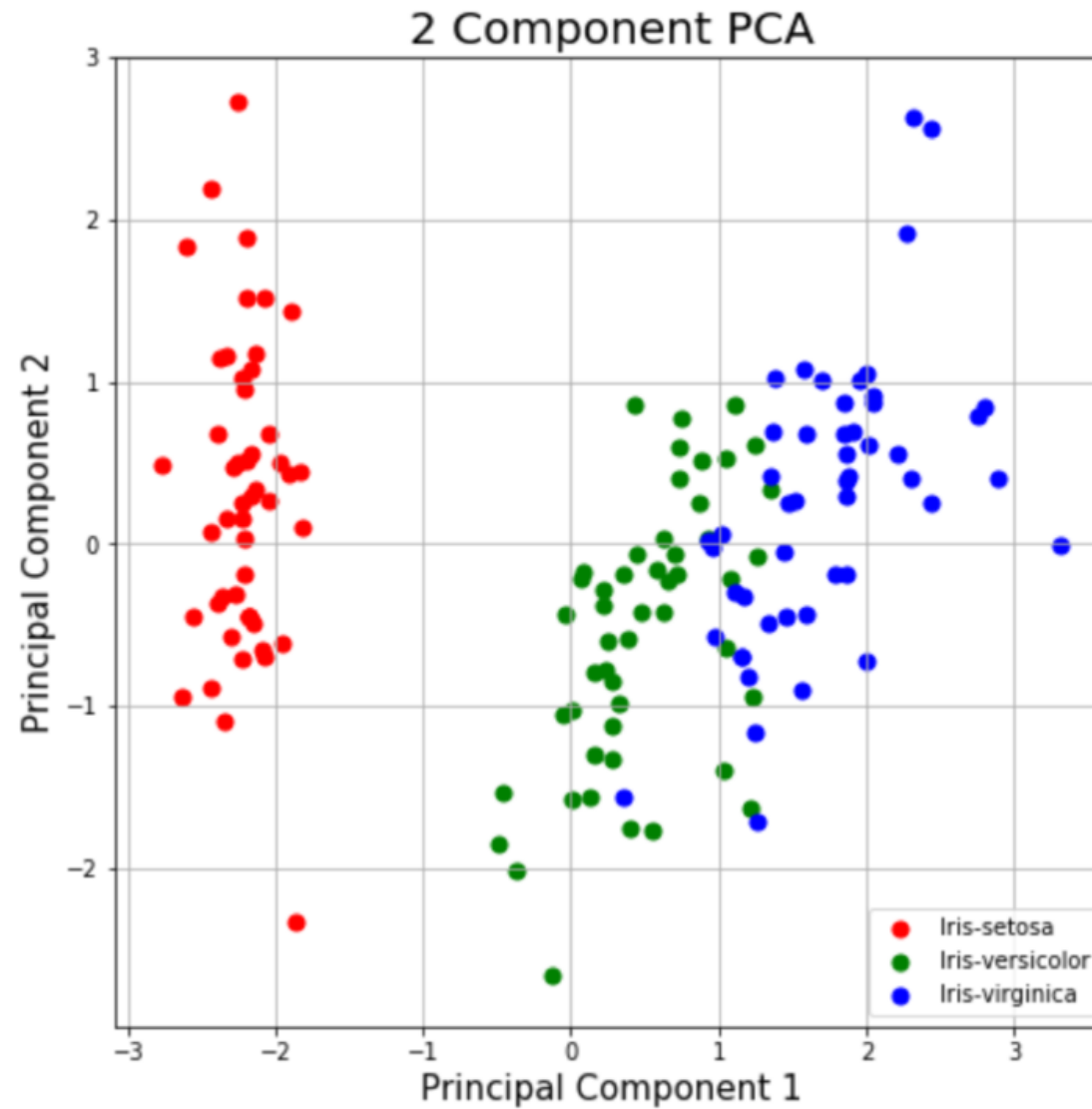
Concatenating dataframes along columns to make finalDf before graphing

Trực quan hóa dữ liệu

- Step 4: Biểu diễn dữ liệu lên biểu đồ

```
from matplotlib import pyplot as plt
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

Trực quan hóa dữ liệu



2 Component PCA Graph

Cảm ơn đã theo dõi!