PCA eigenface

PinkLAB Edu

11 November, 2025

Table of Contents

1 Olivetti 데이터

1.1 데이터

`sklearn.datasets.` **`fetch_olivetti_faces`** (*data_home=None, shuffle=False, random_state=0, download_if_missing=True*) [source]

Load the Olivetti faces data-set from AT&T (classification).

Download it if necessary.

Classes	40
Samples total	400
Dimensionality	4096
Features	real, between 0 and 1

1.2 AT&T와 캠브리지 대학의 공동 연구 데이터

2 코드

2.1 데이터 읽기

```
from sklearn.datasets import fetch_olivetti_faces

faces_all = fetch_olivetti_faces()
print(faces_all.DESCR)
```

✓ 31.4s Python

```
downloading Olivetti faces from https://ndownloader.figshare.com/files/5976027 to
/home/jt/scikit_learn_data
.. _olivetti_faces_dataset:

The Olivetti faces dataset
--------------------------

`This dataset contains a set of face images`_ taken between April 1992 and
April 1994 at AT&T Laboratories Cambridge. The
:func:`sklearn.datasets.fetch_olivetti_faces` function is the data
fetching / caching function that downloads the data
archive from AT&T.
```

이 데이터는 얼굴 인식용으로 사용할 수 있지만,

우리는 특정 인물의 데이터(10장)만 이용해서 PCA 실습용으로 사용

2.2 어떤 모양일까?

```
import numpy as np

print("There are " + str(len(faces_all)) + " images in the dataset")
print(
    "There are "
    + str(len(np.unique(faces_all.target)))
    + " unique targets in the dataset"
)
```

✓ 0.0s Python

```
There are 4 images in the dataset
There are 40 unique targets in the dataset
```

2.3 Label 은 어떤 값들일까?

```
print("unique target number:" + str(np.unique(faces_all.target)))
```

✓ 0.0s Python

```
unique target number:[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39]
```

2.4 데이터를 열어보자

```
import matplotlib.pyplot as plt


def show_40_distinct_people(images, unique_ids):
    fig, axarr = plt.subplots(nrows=4, ncols=10, figsize=(18, 9))
    axarr = axarr.flatten()

    for unique_id in unique_ids:
        image_index = unique_id * 10
        axarr[unique_id].imshow(images[image_index], cmap="gray")
        axarr[unique_id].set_xticks([])
        axarr[unique_id].set_yticks([])
        axarr[unique_id].set_title("face id:{}".format(unique_id))
    plt.suptitle("There are 40 distinct people in the dataset")
```

✓ 0.2s Python

```
show_40_distinct_people(faces_all.images, np.unique(faces_all.target))
```

✓ 0.8s Python

2.5 특정 샘플만 선택

```
K = 20
faces = faces_all.images[faces_all.target == K]
```

✓ 0.0s Python

```
faces
```

✓ 0.0s Python

```
array([[[0.5165289 , 0.5123967 , 0.5082645 , ..., 0.42975205,
         0.42561984, 0.41735536],
        [0.5082645 , 0.5123967 , 0.5206612 , ..., 0.42975205,
         0.42975205, 0.4214876 ],
        [0.4876033 , 0.5123967 , 0.5289256 , ..., 0.4338843 ,
         0.42975205, 0.42975205],
        ...,
```

2.6 무슨 데이터가 있는거지?

```
import matplotlib.pyplot as plt

N = 2
M = 5

fig = plt.figure(figsize=(10, 5))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)

for n in range(N * M):
    ax = fig.add_subplot(N, M, n + 1)
    ax.imshow(faces[n], cmap=plt.cm.bone)
    ax.grid(False)
    ax.xaxis.set_ticks([])
    ax.yaxis.set_ticks([])

plt.suptitle("Olivetti")
plt.tight_layout()
plt.show()
```

✓ 0.2s Python

2.7 Hello~

2.8 두 개의 성분으로 분석

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X = faces_all.data[faces_all.target == K]
W = pca.fit_transform(X)

X_inv = pca.inverse_transform(W)
```

✓ 0.0s Python

2.9 분석된 결과를 확인해보자

```
N = 2
M = 5

fig = plt.figure(figsize=(10, 5))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)
for n in range(N * M):
    ax = fig.add_subplot(N, M, n + 1)
    ax.imshow(X_inv[n].reshape(64, 64), cmap=plt.cm.bone)
    ax.grid(False)
    ax.xaxis.set_ticks([])
    ax.yaxis.set_ticks([])

plt.suptitle("PCA result")
plt.tight_layout()
plt.show()
```

✓ 0.2s Python

2.10 결과

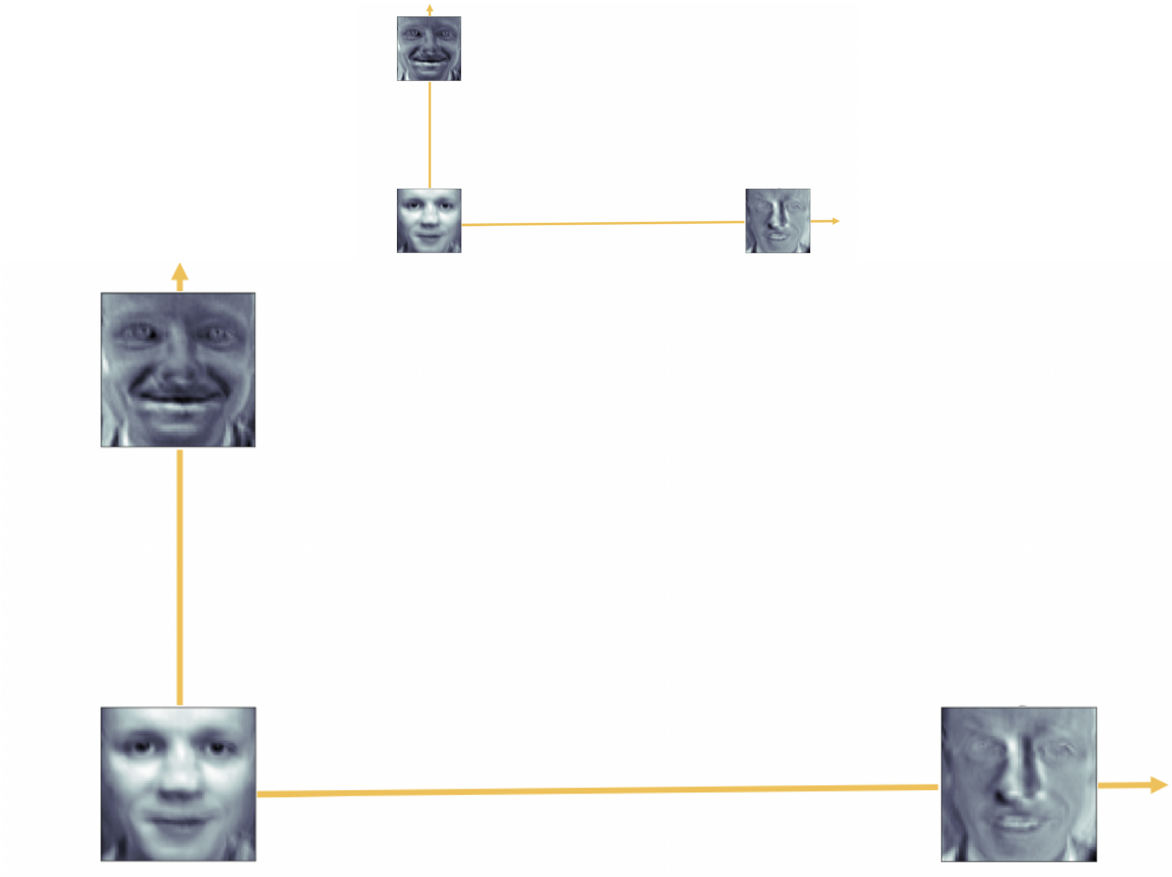
2.11 원점과 두 개의 eigen face

```
face_mean = pca.mean_.reshape(64, 64)

face_p1 = pca.components_[0].reshape(64, 64)
face_p2 = pca.components_[1].reshape(64, 64)


plt.figure(figsize=(12, 7))
plt.subplot(131)
plt.imshow(face_mean, plt.cm.bone)
plt.grid(False)
plt.xticks([])
plt.yticks([])
plt.title("mean")

plt.subplot(132)
plt.imshow(face_p1, cmap=plt.cm.bone)
plt.grid(False)
plt.xticks([])
plt.yticks([])
plt.title("face_p1")

plt.subplot(133)
plt.imshow(face_p2, plt.cm.bone)
plt.grid(False)
plt.xticks([])
plt.yticks([])
plt.title("face_p2")

plt.show()
```

✓ 0.0s Python

2.12 10장의 사진은 이 세장으로 모두 표현할 수 있다

2.13 이렇게 상상해도 된다



2.14 가중치 선정

```python
import numpy as np

N = 2
M = 5
w = np.linspace(-5, 10, N * M)
w
```

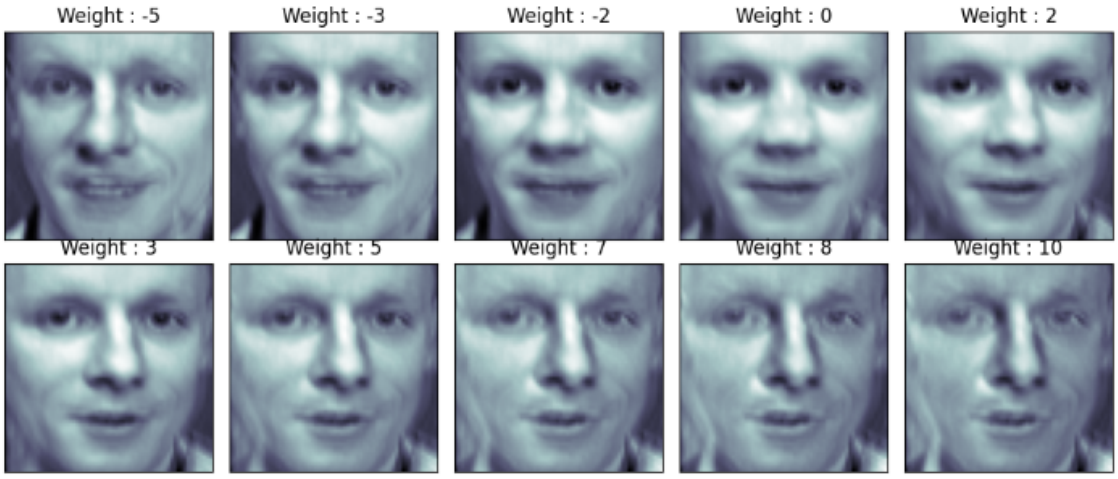
✓ 0.0s Python

```
array([-5.        , -3.33333333, -1.66666667,  0.        ,  1.66666667,
        3.33333333,  5.        ,  6.66666667,  8.33333333, 10.        ])
```

2.15 첫번째 성분의 변화

```python
fig = plt.figure(figsize=(10, 5))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)
for n in range(N * M):
    ax = fig.add_subplot(N, M, n + 1)
    ax.imshow(face_mean + w[n] * face_p1, cmap=plt.cm.bone)
    ax.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.title("Weight : " + str(round(w[n])))


plt.tight_layout()
plt.show()
```

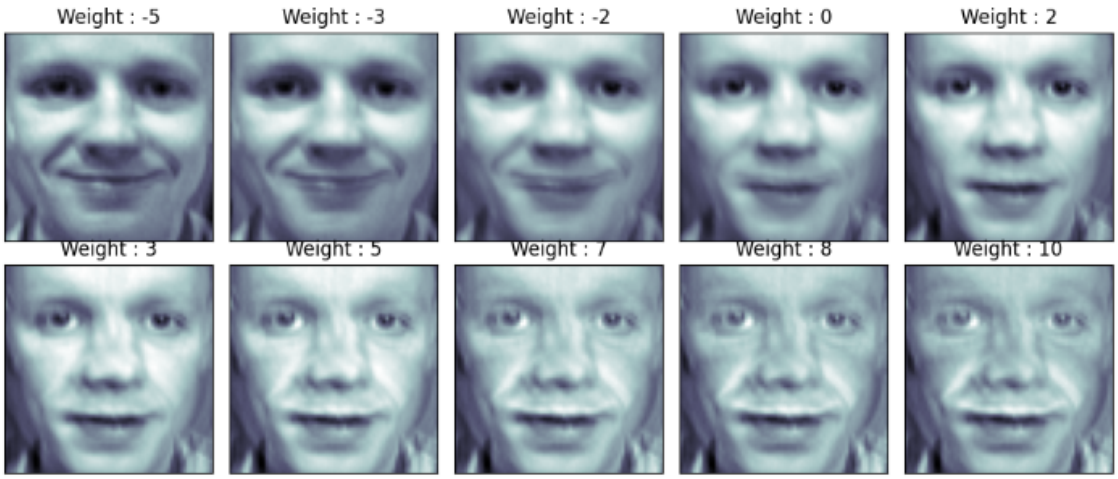
✓ 0.3s Python



2.16 두 번째 성분에 대한 변화

```
fig = plt.figure(figsize=(10, 5))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)
for n in range(N * M):
    ax = fig.add_subplot(N, M, n + 1)
    ax.imshow(face_mean + w[n] * face_p2, cmap=plt.cm.bone)
    ax.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.title("Weight : " + str(round(w[n])))


plt.tight_layout()
plt.show()
```

✓ 0.2s Python



2.17 두 개의 성분을 다 표현해보기

```
nx, ny = (5, 5)

x = np.linspace(-5, 8, nx)
y = np.linspace(-5, 8, ny)
w1, w2 = np.meshgrid(x, y)
w1, w2
```

✓ 0.0s Python

```
(array([[-5.  , -1.75,  1.5 ,  4.75,  8.  ],
        [-5.  , -1.75,  1.5 ,  4.75,  8.  ],
        [-5.  , -1.75,  1.5 ,  4.75,  8.  ],
        [-5.  , -1.75,  1.5 ,  4.75,  8.  ],
        [-5.  , -1.75,  1.5 ,  4.75,  8.  ]]),
 array([[-5.  , -5.  , -5.  , -5.  , -5.  ],
        [-1.75, -1.75, -1.75, -1.75, -1.75],
        [ 1.5 ,  1.5 ,  1.5 ,  1.5 ,  1.5 ],
        [ 4.75,  4.75,  4.75,  4.75,  4.75],
        [ 8.  ,  8.  ,  8.  ,  8.  ,  8.  ]]))
```

2.18 shape 조정

```
w1.shape
```

✓ 0.0s Python

```
(5, 5)
```

```
w1 = w1.reshape(-1,)
w2 = w2.reshape(-1,)
w1.shape
```

✓ 0.0s Python

```
(25,)
```

2.19 다시 합성

```python
fig = plt.figure(figsize=(12, 10))
plt.subplots_adjust(top=1, bottom=0, hspace=0, wspace=0.05)

N = 5
M = 5

for n in range(N * M):
    ax = fig.add_subplot(N, M, n + 1)
    ax.imshow(face_mean + w1[n] * face_p1 + w2[n] * face_p2, cmap=plt.cm.bone)
    ax.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.title("Weight : " + str(round(w1[n], 1)) + ", " + str(round(w2[n], 1)))


plt.tight_layout()
plt.show()
```

✓ 0.7s Python

2.20 결과


Weight : -5.0, -5.0
Weight : -1.8, -5.0
Weight : 1.5, -5.0
Weight : 4.8, -5.0
Weight : 8.0, -5.0
Weight : -5.0, -1.8
Weight : -1.8, -1.8
Weight : 1.5, -1.8
Weight : 4.8, -1.8
Weight : 8.0, -1.8
Weight : -5.0, 1.5
Weight : -1.8, 1.5
Weight : 1.5, 1.5
Weight : 4.8, 1.5
Weight : 8.0, 1.5
Weight : -5.0, 4.8
Weight : -1.8, 4.8
Weight : 1.5, 4.8
Weight : 4.8, 4.8
Weight : 8.0, 4.8
Weight : -5.0, 8.0
Weight : -1.8, 8.0
Weight : 1.5, 8.0
Weight : 4.8, 8.0
Weight : 8.0, 8.0

2.21 뭐 이런 느낌