PCA algorithm

December 3, 2023

[23]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[2]:
```
from sklearn.datasets import load_breast_cancer
```

[3]:
```
#Importing breast cancer dataset from the sk-learn library into the cancer␣
 ↪library
cancer = load_breast_cancer()
#Displaying the keys of the loaded dataset with provides information about the␣
 ↪different attributes of the dataset
cancer.keys()
```

[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

[4]:
```
#Displaying the description of the breast cancer dataset
#The 'DESCR' key typically contains information about the dataset, its␣
 ↪features, and its origin
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
```

```
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

===================================== ====== ======
                                       Min    Max
===================================== ====== ======
radius (mean):                        6.981  28.11
texture (mean):                       9.71   39.28
perimeter (mean):                     43.79  188.5
area (mean):                          143.5  2501.0
smoothness (mean):                    0.053  0.163
compactness (mean):                   0.019  0.345
concavity (mean):                     0.0    0.427
concave points (mean):                0.0    0.201
symmetry (mean):                      0.106  0.304
fractal dimension (mean):             0.05   0.097
radius (standard error):              0.112  2.873
texture (standard error):             0.36   4.885
perimeter (standard error):           0.757  21.98
area (standard error):                6.802  542.2
smoothness (standard error):          0.002  0.031
compactness (standard error):         0.002  0.135
concavity (standard error):           0.0    0.396
concave points (standard error):      0.0    0.053
symmetry (standard error):            0.008  0.079
fractal dimension (standard error):   0.001  0.03
radius (worst):                       7.93   36.04
texture (worst):                      12.02  49.54
perimeter (worst):                    50.41  251.2
area (worst):                         185.2  4254.0
smoothness (worst):                   0.071  0.223
compactness (worst):                  0.027  1.058
concavity (worst):                    0.0    1.252
concave points (worst):               0.0    0.291
```

```
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
```

```
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

[5]:
```python
#Creating a DataFrame (tabular data structure) using the breast cancer dataset␣
 ↪features
#'cancer['data']' contains the feature values, and 'cancer['feature_names']'␣
 ↪contains the column names
df = pd.DataFrame(cancer['data'], columns = cancer['feature_names'])
```

[6]:
```python
#Displaying the first five rows of the new DataFrame
df.head()
```

[6]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
```

```
0                 0.2654          0.4601                  0.11890
1                 0.1860          0.2750                  0.08902
2                 0.2430          0.3613                  0.08758
3                 0.2575          0.6638                  0.17300
4                 0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[7]:
```python
from sklearn.preprocessing import StandardScaler
```

[8]:
```python
#Creating an instance of the StandardScaler, which will be used to standardize␣
↪the data
scaler = StandardScaler()
#Fitting the StandardScaler to the data in the DataFrame 'df'
#This computes the mean and standard deviation necessary for standardization
scaler.fit(df)
```

[8]:
```
StandardScaler()
```

[9]:
```python
#Creating another instance of the StandardScaler class with default parameter␣
↪values which allows you to customize the behavior of the scaler
StandardScaler(copy=True, with_mean=True, with_std=True)
```

[9]:
```
StandardScaler()
```

[10]:
```python
#Using the previously fitted StandardScaler to transform the data in the␣
↪DataFrame 'df'
#The transform method applies the scaling computed during fitting to the input␣
↪data
scaled_data = scaler.transform(df)
```

[11]:
```python
#Displaying the variable 'scaled_data'
scaled_data
```

[11]:
```
array([[ 1.09706398, -2.07333501,  1.26993369, …,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, …,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, …,  1.95500035,
         1.152255  ,  0.20139121],
       …,
       [ 0.70228425,  2.0455738 ,  0.67267578, …,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, …,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, …, -1.74506282,
        -0.04813821, -0.75120669]])
```

[12]:
```
#Importing the Principal Component Analysis (PCA) class from scikit-learn
#PCA is a technique used for dimensionality reduction and feature extraction
from sklearn.decomposition import PCA
```

[14]:
```
#Creating an instance of the PCA (Principal Component Analysis) class
#Set the number of components to 2, indicating that we want to reduce the data␣
↪to two dimensions
pca = PCA(n_components=2)
```

[15]:
```
#Fitting the PCA model to the scaled data
#This step computes the principal components based on the scaled input data
pca.fit(scaled_data)
```

[15]:
```
PCA(n_components=2)
```

[18]:
```
#Transforming the scaled data into the lower-dimensional space using the fitted␣
↪PCA model
#The transform method applies the dimensionality reduction to the input data
x_pca = pca.transform(scaled_data)
```

[16]:
```
#Displaying the shape (dimensions) of the scaled data
#This provides information about the number of samples and features in the␣
↪dataset
scaled_data.shape
```

[16]:
```
(569, 30)
```

[20]:
```
#Displaying the scaled data
#This shows the dataset after standardization, where each feature has zero mean␣
↪and unit variance
scaled_data
```

[20]:
```
array([[ 1.09706398, -2.07333501,  1.26993369, …,  2.29607613,
          2.75062224,  1.93701461],
        [ 1.82982061, -0.35363241,  1.68595471, …,  1.0870843 ,
         -0.24388967,  0.28118999],
        [ 1.57988811,  0.45618695,  1.56650313, …,  1.95500035,
          1.152255  ,  0.20139121],
        …,
        [ 0.70228425,  2.0455738 ,  0.67267578, …,  0.41406869,
         -1.10454895, -0.31840916],
        [ 1.83834103,  2.33645719,  1.98252415, …,  2.28998549,
          1.91908301,  2.21963528],
        [-1.80840125,  1.22179204, -1.81438851, …, -1.74506282,
         -0.04813821, -0.75120669]])
```

[19]:
```
#Displaying the shape (dimensions) of the reduced-dimensional data obtained␣
↪from PCA
#This provides information about the number of samples and the reduced number␣
↪of features/components
x_pca.shape
```

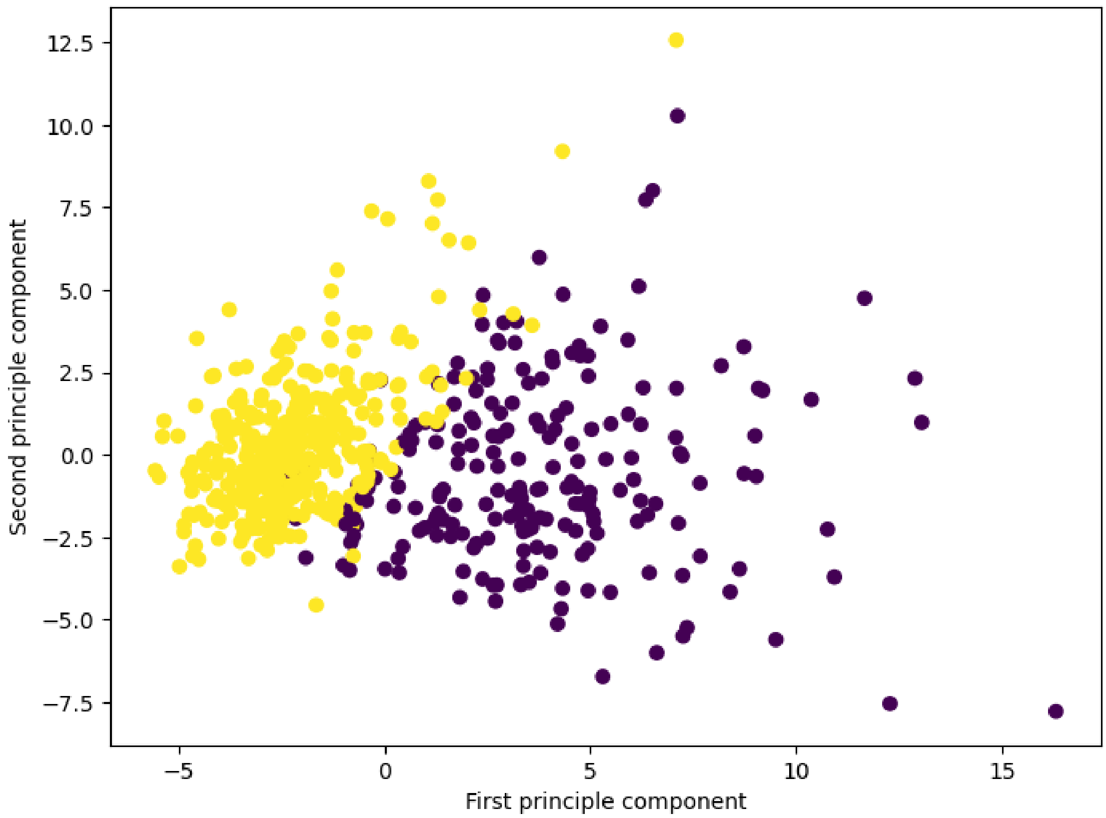
[19]:
```
(569, 2)
```

[21]:
```
#Displaying the reduced-dimensional data obtained from PCA
#This data represents the original dataset transformed into a lower-dimensional␣
↪space
x_pca
```

[21]:
```
array([[ 9.19283683,  1.94858307],
       [ 2.3878018 , -3.76817174],
       [ 5.73389628, -1.0751738 ],
       …,
       [ 1.25617928, -1.90229671],
       [10.37479406,  1.67201011],
       [-5.4752433 , -0.67063679]])
```

[25]:
```
#Creating a scatter plot to visualize the reduced-dimensional data
#Each point in the plot represents a sample, colored based on the target␣
↪variable
#The x-axis corresponds to the first principal component, and the y-axis to the␣
↪second principal component
plt.figure(figsize=(8, 6))
plt.scatter(x_pca[:, 0], x_pca[:, 1], c = cancer['target'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
```

[25]:
```
Text(0, 0.5, 'Second principle component')
```


12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Second principle component
−5
0
5
10
15
First principle component