index

April 20, 2022

1 Performing Principal Component Analysis (PCA) - Lab

1.1 Introduction

Now that you have a high-level overview of PCA, as well as some of the details of the algorithm itself, it's time to practice implementing PCA on your own using the NumPy package.

1.2 Objectives

You will be able to:

- Implement PCA from scratch using NumPy

1.3 Import the data

- Import the data stored in the file `'foodusa.csv'` (set `index_col=0`)
- Print the first five rows of the DataFrame

[3]:

```
import pandas as pd
data = pd.read_csv('foodusa.csv', index_col = 0)
data.head()
```

[3]:

```
           Bread  Burger  Milk  Oranges  Tomatoes
City
ATLANTA     24.5    94.5  73.9     80.1      41.6
BALTIMORE   26.5    91.0  67.5     74.6      53.3
BOSTON      29.7   100.8  61.4    104.0      59.6
BUFFALO     22.8    86.6  65.3    118.4      51.2
CHICAGO     26.7    86.7  62.7    105.9      51.2
```

1.4 Normalize the data

Next, normalize your data by subtracting the mean from each of the columns.

[4]:

```
data = data.mean() - data
data.head()
```

[4]:

```
              Bread    Burger       Milk    Oranges   Tomatoes
City
ATLANTA    0.791304 -2.643478 -11.604348  22.891304   7.165217
```

```
BALTIMORE -1.208696  0.856522  -5.204348  28.391304  -4.534783
BOSTON    -4.408696 -8.943478   0.895652  -1.008696 -10.834783
BUFFALO    2.491304  5.256522  -3.004348 -15.408696  -2.434783
CHICAGO   -1.408696  5.156522  -0.404348  -2.908696  -2.434783
```

1.5 Calculate the covariance matrix

The next step is to calculate the covariance matrix for your normalized data.

```
[6]: cov_mat = data.cov()
     cov_mat
```

```
[6]:              Bread     Burger       Milk     Oranges   Tomatoes
     Bread     6.284466  12.910968   5.719051    1.310375   7.285138
     Burger   12.910968  57.077115  17.507530   22.691877  36.294783
     Milk      5.719051  17.507530  48.305889   -0.275040  13.443478
     Oranges   1.310375  22.691877  -0.275040  202.756285  38.762411
     Tomatoes  7.285138  36.294783  13.443478   38.762411  57.800553
```

```
[18]: covv = data.cov()
      covv
```

```
[18]:             Bread     Burger       Milk     Oranges   Tomatoes
     Bread     6.284466  12.910968   5.719051    1.310375   7.285138
     Burger   12.910968  57.077115  17.507530   22.691877  36.294783
     Milk      5.719051  17.507530  48.305889   -0.275040  13.443478
     Oranges   1.310375  22.691877  -0.275040  202.756285  38.762411
     Tomatoes  7.285138  36.294783  13.443478   38.762411  57.800553
```

1.6 Calculate the eigenvectors

Next, calculate the eigenvectors and eigenvalues for your covariance matrix.

```
[21]: import numpy as np
      eig_values, eig_vectors = np.linalg.eig(cov_mat)
```

```
[21]: array([218.99867893,  91.72316894,   3.02922934,  20.81054128,
              37.66268981])
```

1.7 Sort the eigenvectors

Great! Now that you have the eigenvectors and their associated eigenvalues, sort the eigenvectors based on their eigenvalues to determine primary components!

```
[22]: # Get the index values of the sorted eigenvalues
      e_indices = eig_values.argsort()[::-1]

      # Sort
```

```
eigenvectors_sorted = eig_vectors[:,e_indices]
eigenvectors_sorted
```

```
[22]: array([[-0.02848905, -0.16532108,  0.02135748, -0.18972574, -0.96716354],
              [-0.2001224 , -0.63218494,  0.25420475, -0.65862454,  0.24877074],
              [-0.0416723 , -0.44215032, -0.88874949,  0.10765906,  0.03606094],
              [-0.93885906,  0.31435473, -0.12135003, -0.06904699, -0.01521357],
              [-0.27558389, -0.52791603,  0.36100184,  0.71684022, -0.03429221]])
```

1.8 Reprojecting the data

Finally, reproject the dataset using your eigenvectors. Reproject this dataset down to 2 dimensions.

```
[26]: transformed = eigenvectors_sorted[:2].dot(data.T).T

      transformed.to_frame()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-26-562c18525602> in <module>
      1 transformed = eigenvectors_sorted[:2].dot(data.T).T
      2
----> 3 transformed.to_frame()

AttributeError: 'numpy.ndarray' object has no attribute 'to_frame'
```

1.9 Summary

Well done! You've now coded PCA on your own using NumPy! With that, it's time to look at further applications of PCA.