Performing Principal Component Analysis (PCA) - Lab

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.

Objectives

You will be able to:

- Implement PCA from scratch using NumPy

Import the data

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

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

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	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

Normalize the data

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

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

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	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

Calculate the covariance matrix

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

```
cov_mat = data.cov()
cov_mat
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	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

Calculate the eigenvectors

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

```
import numpy as np
eig_values, eig_vectors = np.linalg.eig(cov_mat)
```

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!

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

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

```
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]])
```

Reprojecting the data

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

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

```
array([[ 11.10636706,  14.73134932],
       [  1.21900301,  21.44989411],
       [-12.29365615,  -4.73286484],
       [ -4.27410508,  -4.95746329],
       [ -2.08570219,   1.77071529],
       [  2.02322957,  -4.83130671],
       [  0.76513968,  -5.78241861],
       [  5.01990281,  -7.08302707],
       [ -5.28963606,  -7.18560259],
       [-20.29491244, -21.8073279 ],
       [  6.65984778,   1.94000258],
       [  6.31128707,   2.48929146],
       [ 14.12127203,  10.53050299],
       [ -6.30360841,  -4.25846371],
       [ -2.70119271,  -4.22170394],
       [-17.39101398, -11.4694373 ],
       [-11.5197783 ,   7.50279463],
       [ -3.85046231, -12.11400365],
       [  0.02742717,  -9.57890292],
       [ 16.08930141,   7.15585369],
       [  7.92495397,   0.76667379],
       [  8.40796253,  16.30333807],
       [  6.32837355,  13.38210659]])
```

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.

Releases

No releases published

Packages

No packages published

Contributors 7

Languages

- **Jupyter Notebook** 100.0%