Image Analysis

Exercise 1b

Principal Component Analysis (PCA) in Python

Fall 2022

Introduction

This exercise is an introduction to principal component analysis in Python. The classical iris data set is used as an example. Check here: `https://en.wikipedia.org/wiki/Iris_flower_data_set` for an explanation.

Learning Objectives

After completing this exercise, the student should be able to do the following:

1. Use NumPy to read text files.
2. Create a data matrix from a text file
3. Organise measured data into a data matrix
4. Compute the variance of a set of measurements
5. Compute the covariance between two sets of measurements
6. Use the function `pairplot` from the `seaborn` package to visualise the covariance between multiple sets of measurements
7. Compute the covariance matrix from multiple sets of measurements using matrix multiplications
8. Compute the covariance matrix from multiple sets of measurements using the NumPy `cov` function.
9. Compute the principal components using Eigenvector analysis (NumPy function `eig`).
10. Visualize how much of the total of variation each principal component explain.

11. Project original measurements into principal component space
12. Use the function `pairplot` to visualise the covariance structure after projecting to principal component space.
13. Compute the PCA using the `PCA` function from the `sci-kit learn decomposition` package.

Getting the Data

The data and material needed for this exercise can be downloaded from

`http://courses.compute.dtu.dk/02502/`

Start by creating an exercise folder where you keep your Python data. Download the data and place them in this folder.

Loading the data

We start by reading the data and in this example we only use the first 50 measurements from the same type of flower:

Exercise 1

Start by reading the data create a data matrix **x**:

```
import numpy as np
iris_data = np.loadtxt(in_dir + txt_name, comments="%")
# x is a matrix with 50 rows and 4 columns
x = iris_data[0:50, 0:4]
```

then check the data dimensions by writing:

```
n_feat = x.shape[1]
n_obs = x.shape[0]
print(f"Number of features: {n_feat} and number of observations: {n_obs}")
```

We have 50 flowers and for each flower there are 4 measurements (features): sepal length, sepal width, petal length and petal width. Are your matrix dimensions correct?

Explorative data analysis

Exercise 2

To explore the data, we can create vectors of the individual feature:

```
sep_l = x[:, 0]
sep_w = x[:, 1]
pet_l = x[:, 2]
pet_w = x[:, 3]
```

Compute the variance of each feature like:

```
# Use ddof = 1 to make an unbiased estimate
var_sep_l = sep_l.var(ddof=1)
```

do that for all the features.

Exercise 3

Now compute the covariance:

$$\sigma_{AB}^2 = \frac{1}{N-1} \sum_i a_i b_i$$

between the sepal length and the sepal width. Please note that we use $N-1$ instead of just N in the computation of the covariance. The reason for this can be found in estimation theory. Also note that the covariance we compute is not equal to for example the `Numpy cov` function. It can, though, still tell us something about the data.

Compute the covariance between the sepal length and the petal length and compare it to the covariance between the sepal length and width. What do you observe?

Exercise 4

As with image analysis, it is very useful to get a graphical understanding of the data and what structures are hidden in the them. For this, we will use the `seaborn` Python package that also used the `pandas` package. Take a look at Appendix A for installation instructions.

Import seaborn and pandas:

```
import seaborn as sns
import pandas as pd
```

Now let us get a closer look at the data structure using a `pairplot`:

```
plt.figure() # Added this to make sure that the figure appear
# Transform the data into a Pandas dataframe
d = pd.DataFrame(x, columns=['Sepal length', 'Sepal width',
                             'Petal length', 'Petal width'])
sns.pairplot(d)
plt.show()
```

What measurements are related and which ones are not-related? Can you recognise the results you found, when you computed the variance and covariance?

PCA Analysis

We will do the principal component analysis (PCA) in two ways. First, a method combining several steps and finally using a dedicated pca library that does all at once.

Exercise 5

In the first approach, we do the analysis step-wise.

Start by subtracting the mean from the data:

```
mn = np.mean(x, axis=0)
data = x - mn
```

Now compute the covariance matrix using:

$$\mathbf{C_X} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}.$$

Remember to use the data, where the mean has been subtracted. You can use the NumPy function `matmul` to multiply two matrices together. Also remember to transpose data in one of the arguments to the function.

You can also use the NumPy function cov to compute the covariance matrix. Verify if the two approaches give the same result?

Exercise 6

Now you can compute the principal components using eigenvector analysis:

```
values, vectors = np.linalg.eig(c_x)
```

Here `c_x` is your covariance matrix.

The `values` are the eigenvalues and the `vectors` are the eigenvectors (the principal components).

Exercise 7

Lets examine some properties of the principal components.

First try to find out how much of the total variation the first component explains?

You can also plot the amount of explained variation for each component:

```
v_norm = values / values.sum() * 100
plt.plot(v_norm)
plt.xlabel('Principal component')
plt.ylabel('Percent explained variance')
plt.ylim([0, 100])
plt.show()
```

Exercise 8

The data can be projected onto the PCA space by using the dot-product:

```
pc_proj = vectors.T.dot(data.T)
```

Try to use seaborns pairplot with the projected data? How does the covariance structure look like?

Direct PCA using the decompositions functions

The Python machine learning package `sci-kit learn (sklearn)` have several functions to do data decompositions, where PCA is one of them.

Let us try to see if we get the same results using these function.

Start by installing sklearn (see Appendix A) and importing the package:

```
from sklearn import decomposition
```

Exercise 9

Read the data matrix as before, but do not subtract the mean. The procedure subtract the mean for you.

The PCA can be computed using:

```
pca = decomposition.PCA()
pca.fit(x)
values_pca = pca.explained_variance_
exp_var_ratio = pca.explained_variance_ratio_
vectors_pca = pca.components_

data_transform = pca.transform(data)
```

Compare the results from the results you found using the step-by-step procedure. Some of the results are transposed - which ones?

A Package installations

sklearn (sci-kit learn)

(Further info: `https://anaconda.org/anaconda/scikit-learn`, `https://scikit-learn.org/stable/`)

```
activate course02502
conda install -c anaconda scikit-learn
```

seaborn

(Further info: `https://anaconda.org/anaconda/seaborn`, `https://seaborn.pydata.org/`)

```
activate course02502
conda install -c anaconda seaborn
```