Practical Session 7, 2019/2020(B)

1. Aim of the Session

   In this session, you will learn how to do the Principal Component Analysis in Python.

2. Understand the PCA through A Simple Example

   Task1:

   1) Open a new notebook in jupyter notbook.
   2) Create an array including 5 data points using the **NumPy** module:

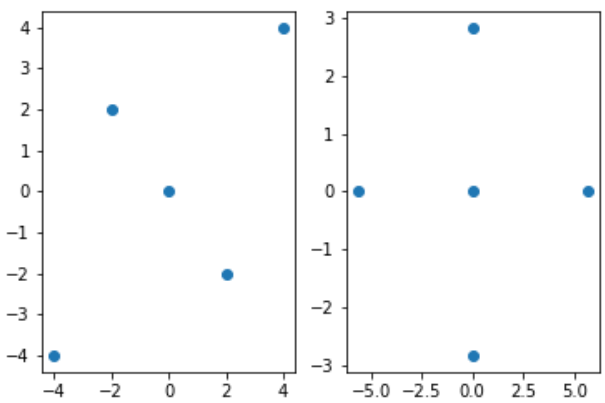
   $$X = \begin{pmatrix} -2 & 2 \\ -4 & -4 \\ 4 & 4 \\ 2 & -2 \\ 0 & 0 \end{pmatrix}$$

   3) Plot a scatter plot using the scatter from the **Matplotlib.pyplot** module.
   4) Computer the covariance matrix (sigma) of X using the **cov()** function from the **NumPy** module. Check the sum along the main diagonal line of sigma.
   5) Do the eigendecomposition on sigma using the following code:

      ```
      from numpy import linalg as LA
      eignVal, eignVec = LA.eig(sigma)
      ```

      Check if the sum of the eignVal is equal to the sum along the main diagonal line of sigma.
   6) Project X onto the eigenvectors to get projections (proj_x).
   7) Plot a scatter plot of proj_x.

      You should get a figure similar to the following: the left panel shows the original data; the right panel shows projections of the data.



8) Reflection: through this exercise, what conclusions have you got?

3. Principal Component Analysis Using Scikit-Learn

You can import PCA from Scikit-Learn as follows:

```
from sklearn.decomposition import PCA

pca = PCA(n_components =2)
proj_X_2 = pca.fit_transform(X)
```

Task2:

1) Plot a scatter plot of proj_X_2 and compare it with the one you have got in Task 1. Are they the same?
2) You can access the principal components using the components_ variable. For example, run the following code

```
pca.components_
```

Find more details about the function PCA using **?PCA.**

4. SVD and PCA

Task3: Given $X = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix}$, find an SVD of X

1) The transpose of X
2) Compute $M = X^T X$
3) Find the eigenvalue and eigenvectors of M
4) What is the singular value matrix S and what is the right singular matrix V?
5) Set up the left singular matrix U.
6) Compare your results with the results obtained using the **svd** function in Python, that is, using 'from numpy.linalg import svd'.

5. Do a PCA analysis on the wine dataset

Task4:

1) Load wine data

```
from sklearn.datasets import load_wine
```

2) Investigate data: how many data points are there? How many features are there? And what are they? How many classes are included? Plot the boxplot for each feature.
3) Pre-processing the data as follows:

```
from sklearn.preprocessing import StandardScaler
x = StandardScaler().fit_transform(data)
```

4) Do PCA on the normalised wine dataset (**x**).

Project the normalised wine data on the first two principal components. Plot the PCA visualisation plot using the scatter function. Label the data using their class label information. How much variance has been captured by the first two principal components?

5) Plot the scree plot. To keep 80% of the total variance, how many principal components should be used?

6. Naïve Bayes Classification of the Wine Dataset
   Task5:
   1) Divide the wine dataset into a training set (70%) and a test set (30%)
   2) Pre-processing the training set and the test set (Note that the parameters should be obtained from the training set only.)

      *"The `preprocessing` module further provides a utility class **`StandardScaler`** that implements the `Transformer` API to compute **the mean and standard deviation on a training set** so as to be able to later reapply the same transformation on the testing set."*
      https://scikit-learn.org/stable/modules/preprocessing.html
   3) Do a Naïve Bayes Classification using the original features.
   4) Use features extracted from PCA using the first five principal components.

7. Learn more from a nice post on how to use PCA, which can be found from the following link:

   https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

References

[1] J. VanderPlas: Python Data Science Handbook, 2016. O'Relly Media, Inc