Homework Assignment 1:

Linear Algebra
Data and feature extraction
Measures of similarity

Objective: The objective is to get acquainted with the Python language, with emphasis on its scientific and numerical extensions.how data can be imported from other data sources, linear algebra basics that needed in this course, visualized using principal component analysis (PCA) and similarity concept. Upon completing this exercise it is expected that you:

- Understand how data can be represented as vectors and matrices in numerical Python (NumPy).
- Can apply and interpret principal component analysis (PCA) for data visualization.
- Understand the various measures of similarity such as Jaccard and Cosine similarity and apply similarity measures to query for similar observations.

Materials: The following on-line materials are recommended.

- `docs.python.org/tutorial` - Introduction into Python environment, syntax and data structures. Recommended reading - sections 1, 2, 3, 4 and 5.
- `www.scipy.org/Tentative_NumPy_Tutorial` - Tutorial introducing the scientific computing in Python, array and matrix operations, indexing and slicing matrices.
- `www.scipy.org/NumPy_for_Matlab_Users` - Useful reference to scientific computing in Python if you have previous experience with MATLAB programming.
- `matplotlib.sourceforge.net` - Documentation and examples related to matplotlib module, which we shall use extensively through the course to visualize data and results.
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, sections 2.1-2.3 + (A) + B.1
- `http://www.youtube.com/course?list=ECEA1FEF17E1E5C0DA` - Series of video tutorials covering basics of Python programming.

- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, sections 2.4 + 3.1-3.2 + C.1-C.2

Python help: You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object. Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, often the fastest way is to google your problem, or refer to on-line documentation of packages that you are using (the links will be provided).

Additional Package For this homework, you need to install the following packages or modules, all of them are provided on the Blackboard:

- package *xlrd*, for Windows user, the installer can be downloaded at https://pypi.python.org/pypi/xlrd/0.7.9
- module *similarity.py* (run the script to install it)

Important: The following points is how you hand-in the homework assignment.

- Provide clear and complete answers to the questions below (not hidden somewhere in your source code), and make sure to explain your answers / motivate your choices. Please make as PDF file.
- Source code, output graphs, derivations, etc., should be included, and zipped together with the PDF file.
- Hand-in: upload to Blackboard.
- Include name, student number, assignment (especially in filenames)
- For problems or questions: use the BB discussion board or email.

1.1 Python and Linear Algebra Basics

1.1.1 Generate the following vectors using functions from *NumPy* package in Python :

$$\mathbf{x} = \begin{pmatrix} 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{pmatrix} \mathbf{y} = \begin{pmatrix} 3 \\ 7 \\ 11 \\ 15 \\ 19 \\ 23 \\ 27 \end{pmatrix} \mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0.5 \\ 1 \\ 1.5 \\ 2 \\ 0 \\ 0 \end{pmatrix} \mathbf{s} = \begin{pmatrix} 100 \\ 98.8 \\ 97.6 \\ 96.4 \\ 95.2 \end{pmatrix} \mathbf{z} = \begin{pmatrix} 0.7 \\ 1.0 \\ 1.3 \\ 1.6 \\ 1.9 \\ 2.2 \\ 2.5 \\ 2.8 \end{pmatrix}$$

and compute several operations as follows:

a. $\mathbf{v} = 3\mathbf{x} + \mathbf{y}$
b. compute the dot product between $\mathbf{x}$ and $\mathbf{y}$
c. $\mathbf{t} = \text{pi}(\mathbf{s} + 4)$ (element wise multiplication)
d. $\mathbf{z} = \mathbf{z} - 1$
e. *replace* some values of vector $\mathbf{x}$, such that its last three values = 4
f. $\mathbf{r} = 2\mathbf{w} - 5$

Hints:

- To get description of function, you can simply type `help(obj)` in the Python interpreter or complete *NumPy* documentation can be found in *Spyder* under tab "?"

1.1.2 In Python, generate these matrices. and compute the following operations, if some operations turn to error, give the reason you think why that happens:

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 3 \\ 6 & 8 & 4 \\ 6 & 7 & 5 \end{pmatrix} \mathbf{N} = \begin{pmatrix} 4 & 6 \\ 7 & 2 \\ 5 & 1 \end{pmatrix} \mathbf{P} = \begin{pmatrix} 2 & 5 \\ 5 & 5 \end{pmatrix}$$

a. $\mathbf{A} = \mathbf{MN} + \mathbf{N}$
b. $\mathbf{B} = \mathbf{N}^{\mathsf{T}}\mathbf{M}$

c. $\mathbf{C} = \mathbf{P}^{-1} + \mathbf{P}$

d. $\mathbf{AC}(\mathbf{C} + \mathbf{B})$

e. compute the eigenvalues and eigenvectors of $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{P}$

Hints:

- Employ functions from *NumPy* for matrix operations

1.2 Principal Component Analysis

1.2.1 Many experimenters have a habit of using Microsoft Excel as their tool to record measurements from experiments. Fortunately Python can read Excel files, although it usually needs a bit of tweaking.

The data used in this exercise is based on data from a chemical sensor obtained from the NanoNose project [1]. The data contains 8 sensors named by the letters A-H measuring different levels of concentration of Water, Ethanol, Acetone, Heptane and Pentanol injected into a small gas chamber. The data will be represented in matrix form such that each column contains the 8 sensors measurements (i.e., sensor A-H) of the various compounds injected into the gas chamber.

a. Inspect the file `nanonose.xls` and make sure you understand how the data is stored in Excel.
Write in Python how to load the data into Python using the `xlrd` package, and get it into the standard data matrix form as described in the beginning of this document.

Hints:

- You can read data from excel spreadsheets after installing and importing `xlrd` module. In most cases, you will need only few functions to accomplish it: `open_workbook()`, `sheet_by_index()`, `col_values()`, `row_values()`.
- If you need more advanced reference, or if you are interested how to write data to excel files, see the following tutorial: `http://www.simplistix.co.uk/presentations/python-excel.pdf`.

There are 90 data objects with 8 attributes each. Can you get the correct data matrix $\mathbf{X}$ of size 90×8?

b. The data resides in an 8 dimensional space where each dimension corresponds to each of the 8 NanoNose sensors. This makes visualization of the raw data difficult, because it is difficult to plot data in more than 2 or 3 dimensions.
In Python, plot the two attributes A and B against each other in a scatter plot.

Hints:

- You need to import the `pylab` package to use plotting functions in Python: `from pylab import *`
- In plotting, convert the chosen matrix columns for x and y respectively, into array using *numpy.array()*
- You can find extensive help and numerous examples on the `matplotlib` website: `http://matplotlib.sourceforge.net`.

Try to change with different attributes that are plotted against each other.

1.2.2 We will use principal component analysis (PCA) to reduce the dimensionality of the data. To compute a PCA we first subtract the mean of the data, $\mathbf{Y} = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is a (row) vector containing the mean value of each attribute and $\mathbf{1}$ is an $N \times 1$ column vector of ones in all entries. Next we calculate the so-called singular value decomposition (SVD) of the zero mean data, i.e., $\mathbf{Y} = \mathbf{USV}^T$.

From PCA we can find out how much of the variation in the data each PCA component accounts for. For component m, this is given by

$$\rho_m = 1 - \frac{\|\mathbf{Y} - \mathbf{u}_m s_{mm} \mathbf{v}_m^T\|_F^2}{\|\mathbf{Y}\|_F^2} = \frac{s_{mm}^2}{\sum_{m'=1}^{M} s_{m'm'}^2}$$

i.e., the squared singular value of the given component divided by the sum of all the squared singular values.

a. Using Python, compute the PCA of the NanoNose data and plot the percent of variance explained by the principal components.

Hints:

- You cannot directly subtract a vector from a matrix. One way to accomplish this is to subtract the product of vector of ones and vector of means: `Y = X - np.ones((N,1))*X.mean(0)`.
- You can use the function `numpy.linalg.svd()` to compute the SVD.
- To extract the diagonal from a matrix, use the method `diagonal` of the matrix object.

Can you verify that more than 90% of the variation in the data is explained by the first 3 principal components?

b. In Python, plot principal component 1 and 2 against each other in a scatterplot.

   Hints:

   - Data can be projected onto the principal components using `Z = Y*V` or equivalently computed directly from the SVD as `Z = U*S`.

   What are the benefits of visualizing the data by the projection given by PCA over plotting two of the original data dimensions against each other?

c. Which of the original attributes does the second principal component mainly capture the variation of and what would cause an observation to have a large negative/positive projection onto the second principal component?

   Hints:

   - The columns of `V` give you the principal component directions.
   - The data is projected onto the second principal component by `Y*V[:,1]`.

1.3 Measures of similarity

We will use a subset of the data on wild faces described in [2] transformed to a total of 1000 gray scale images of size 40×40 pixels, we will attempt to find faces in the data base that are the most similar to a given query face. To measure similarity we will consider the following measures: SMC, Jaccard, Cosine, ExtendedJaccard, and Correlation. These measures of similarity are described in *Introduction to Data Mining*, page 73-77 and are given by

$$\begin{aligned}
\text{SMC}(\mathbf{x}, \mathbf{y}) &= \frac{\text{Number of matching attribute values}}{\text{Number of attributes}} \\
\text{Jaccard}(\mathbf{x}, \mathbf{y}) &= \frac{\text{Number of matching presences}}{\text{Number of attributes not involved in 00 matches}} \\
\text{Cosine}(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \\
\text{ExtendedJaccard}(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \mathbf{x}^T\mathbf{y}} \\
\text{Correlation}(\mathbf{x}, \mathbf{y}) &= \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\text{std}(\mathbf{x})\text{std}(\mathbf{y})}
\end{aligned}$$

where $\mathrm{cov}(\mathbf{x}, \mathbf{y})$ denotes the covariance between $\mathbf{x}$ and $\mathbf{y}$ and $\mathrm{std}(\mathbf{x})$ denotes the standard deviation of $\mathbf{x}$.

Notice that the SMC and Jaccard similarity measures only are defined for binary data, i.e., data that takes values in $\{0, 1\}$. As the data we analyze is non-binary, we will transform the data to be binary when calculating these two measures of similarity by setting

$$x_i = \begin{cases} 0 & \text{if } x_i < \mathrm{median}(\mathbf{x}) \\ 1 & \text{otherwise.} \end{cases}$$

1.3.1 Inspect and run the script `ex3_2_1.m`. The script loads the CBCL face database, computes the similarity between a selected query image and all others, and display the query image, the 5 most similar images, and the 5 least similar images. The value of the used similarity measure is shown below each image. Try changing the query image and the similarity measure and see what happens. Which similarity measures produce similar results? Which one gives the best result? and argue why?

1.3.2 We will investigate how scaling and translation impact the following three similarity measures: Cosine, ExtendedJaccard, and Correlation. Let α and β be two constants. Using Python, calculate the following similarity measures, and check if the statements below are correct.

$$\begin{aligned} \mathrm{Cosine}(\alpha\mathbf{x}, \mathbf{y}) &= \mathrm{Cosine}(\mathbf{x}, \mathbf{y}) \\ \mathrm{ExtendedJaccard}(\alpha\mathbf{x}, \mathbf{y}) &= \mathrm{ExtendedJaccard}(\mathbf{x}, \mathbf{y}) \\ \mathrm{Correlation}(\alpha\mathbf{x}, \mathbf{y}) &= \mathrm{Correlation}(\mathbf{x}, \mathbf{y}) \\ \mathrm{Cosine}(\beta + \mathbf{x}, \mathbf{y}) &= \mathrm{Cosine}(\mathbf{x}, \mathbf{y}) \\ \mathrm{ExtendedJaccard}(\beta + \mathbf{x}, \mathbf{y}) &= \mathrm{ExtendedJaccard}(\mathbf{x}, \mathbf{y}) \\ \mathrm{Correlation}(\beta + \mathbf{x}, \mathbf{y}) &= \mathrm{Correlation}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

Hints:

- Type `help similarity` to learn about the function that is used to compute the similarity measures.
- Even though a similarity measure is theoretically invariant e.g. to scaling, it might not be exactly invariant numerically.

References

[1] Tommy S Alstrøm, Jan Larsen, Claus H Nielsen, and Niels B Larsen. Data-driven modeling of nano-nose gas sensor arrays. In *SPIE Defense, Security, and Sensing*,

pages 76970U–76970U. International Society for Optics and Photonics, 2010. URL http://www.nanonose.dk.

[2] Tamara L Berg, Alexander C Berg, Jaety Edwards, and DA Forsyth. Who's in the picture. *Advances in neural information processing systems*, 17:137–144, 2005.