Data and feature extraction with PYTHON

Objective: To get acquainted with the way data can be represented in Python, how data can be imported from other data sources, and how data can be filtered and visualized using principal component analysis (PCA). Upon completing this exercise it is expected that you:

- Understand how data can be represented as vectors and matrices in numerical Python (NumPy)
- Can import data into Python from Excel and Matlab `.mat` files and represent them in course $\mathbf{X}$, $\mathbf{y}$ format.
- Can apply and interpret principal component analysis (PCA) for data visualization.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* C2, C3 as well as the files in the exercise 2 folder available from Campusnet.

Preparation: Exercise 1

Part 1: Group discussion (max 15 min)

For the group discussion, each group should have selected a *discussion leader* at the previous exercise session. The purpose of the discussion leader is to ensure all team members understands the answers to the following two questions:

Multiple-Choice question: Solve and discuss **problem 3.1** from chapter 3 of the lecture notes. Ensure all group members understand the reason why one of the options is true and why the other options can be ruled out. (After today's exercises make sure to complete the remaining multiple-choice problems listed as part of the preparation for week 2 on the course homepage).

Discussion question: Discuss the following question in the group

- Consider the slide "Visualization of hand written digits". Explain how the plots are produced from a collection of images. In particular, explain what each $\mathbf{v}_1, \mathbf{v}_2$ correspond to, i.e. how are they computed and what are their dimensions? Explain what each of the black points correspond to.

Part 2: Programming exercises

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, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

Piazza discussion forum: You can get help by asking questions on Piazza:
https://piazza.com/dtu.dk/fall2017/02450

Software installation: Extract the Python toolbox from Campusnet. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` For today's exercises you need to add few more packages to your Python environment. The additional packages are related to natural language processing, machine learning, and importing data from excel spreadsheets. Please make sure that you have installed the following packages (you can follow the guidelines at the corresponding websites):

- Natural Language Toolkit (nltk package) - powerful package for natural language processing - stemming, tokenizing, parsing, semantic analysis:
  `http://www.nltk.org/install.html`

- Excel file data extraction (xlrd package):
  `http://www.lexicon.net/sjmachin/xlrd.html`

  And for the last exercise (today optionally, but we shall need it in the following weeks):

- Machine learning toolkit (scikit-learn) - large package implementing various ML methods for supervised and unsupervised learning:
  `http://scikit-learn.org/stable/install.html`

The websites provide documentation of the packages. Note if you use the Anaconda Python distribution these packages may already be added, use `conda list` in the terminal for a list of installed packages.

Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

2.1 PCA on the Nanose dataset

In this section, we will consider PCA for the Nanose dataset already encountered in last weeks exercises. The nanose dataset was obtained from the NanoNose [1] project, see also [2]. 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 row contains the 8 sensors measurements (i.e. sensor A-H) of the various compounds injected into the gas chamber.

2.1.1

Load the Nanose dataset from the file `<base-dir>/02450Toolbox_Python/Data/nanonose.xls` into into Python using the `xlrd` package, and get it into the standard data matrix form as described in the beginning of this document. See `ex2_1_1.py` for details.

Script details:

- *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()`, `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`

Look at the solution in `ex2_1_1.py` to see how the attribute names and class names and indices can be extracted. Run the script and look at the variable that it returns.

2.1.2

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 2–3 dimensions.

Plot the two attributes A and B against each other in a scatter plot using `ex2_1_2.py`.

Script details:

- *You need to import* `matplotlib.pyplot` *package to use plotting functions in Python:*
  `from matplotlib.pyplot import *`
- *Use* `plot()` *function to plot data.*
- *The attributes A and B are the first and second columns of the matrix $\boldsymbol{X}$.'*
- *You can use indexing to get the columns out of the matrix, e.g.,* `x=X[:,1]` *or* `y = X[:,2]`
- *Notice that the third argument of the* `plot()` *command can be used to set a plot symbol. For example, the command* `plot(x,y,'o')` *plots a scatter plot with circles.*
- *Use* `show()` *function to render the plot.*
- *You can find extensive help and numerous examples on matplotlib website:* `http://matplotlib.sourceforge.net`

Try to change the dimensions that are plotted against each other.

We will use principal component analysis to reduce the dimensionality of the data. PCA is computed by subtracting the mean of the data, $\boldsymbol{Y} = \boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}$ (where $\boldsymbol{\mu}$ is a (row) vector containing the mean value of each attribute and $\mathbf{1}$ is a N by 1 column vector of ones in all entries) and then calculating the singular value decomposition (SVD) of the zero mean data, i.e. $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$.

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

$$\rho_m = \frac{s_{mm}^2}{\sum_{m=1}^{M} s_{m,m}^2},$$

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

2.1.3 Compute the PCA of the NanoNose data and plot the percent of variance explained by the principal components using `ex2_1_3.py`.

Script details:

- *You can use the method* `mean()` *of array or matrix object to compute the mean of the data. You should compute the mean for each attribute(column), i.e., the vector of means should have M elements.*
- *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 an array object, or use* `np.diagonal()` *or* `np.diag()`.

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

2.1.4 Plot principal component 1 and 2 against each other in a scatterplot, see the script `ex2_1_4.py` for details.

Script details:

- *Data can be projected onto the principal components using* `Z = Y@V` *or* `Z = np.dot(Y,V)`, *where* `Y` *is centered data.*
- *You learned how to make a scatter plot in Exercise 2.1.2.*

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? Compare with the scatter plots of attributes you made in Exercise 2.1.2.

2.1.5 Consider the script `ex2_1_5.py`. 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?

Script details:

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

2.2 Structure in handwritten digits

The US Postal Service (USPS) wanted to automate the process of sorting letters based on their zip-codes. We will presently consider a dataset of USPS handwritten digits available at `http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html`, see also [3].

2.2.1 Load from zipdata.mat two datasets containing handwritten digits `testdata` and `traindata`.
Note how you can load the matlab data (matlab's m-file) to Python environment with `loadmat()` function imported from `scipy.io` module. The matlab workspace is loaded as dictionary, with keys corresponding to matlab variable names, and values to arrays representing matlab matrices.

2.2.2 Inspect and run the script `ex2_2_1.py` to visualize the first digit of the traindata (the script uses `reshape` to turn a digit vector into an image and `imshow()` to display the image).

2.2.3 Inspect and run the script `ex2_2_2.py`. Show that it requires 22 PCA components to account for more than 90% of the variance in the data. Show that the first principal component is almost sufficient to separate zeros and ones. Examine the first principal component and discuss and understand what it captures.

2.2.4 Change the value of `K` and show that reconstruction accuracy improves when more principal components are used. How many principal components do you need to be able to see the different digits properly? What happens if you set `K=256`?

2.2.5 Try decomposing one digit at a time. Hint: Modify the variable `n` to contain only a single digit. Explain what happens to the principal components when only a single digit type is analyzed compared to when all digit types are analyzed at the same time.

2.3 Extra challenge

We will later in the course learn various methods for classification. Among the approaches we will learn is K-nearest neighbor (KNN) classification. For now we will consider the KNN classifier a black box that we will use to evaluate how well we can determine the digit class in the space given by the K first principal components, i.e. after filtering out the PCA components with smallest singular values which we consider components pertaining to noise.

2.3.1 Inspect and run the script `ex2_3_1.py` and see how well we are able to classify the digits when we use say `K=10` PCA components, `K=40` PCA components and the whole data, i.e `K=256` PCA components. Show that the classifier is best when using around 40–60 PCA components, and explain why that is so.

2.4 Tasks for the report

After today's exercise you should be able to load your data into Python and put it in the format described in the section on "Representation of data in Python" at the beginning of today's exercise. You should also be able to explain what types of attributes are in your data (i.e. discrete/continous, Nominal/Ordinal/Interval/Ratio, see also today's lecture) as well as be able to apply and interpret the results of a principal component analysis (PCA) of your data. Notice, that there are three aspects that needs to be described in the PCA analysis for the report:

- The amount of variation explained as a function of the number of PCA components included,
- the principal directions of the considered PCA components,
- the data projected onto the considered principal components.

References

[1] Nanonose project.

[2] 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.

[3] Jonathan J. Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5):550–554, 1994.