# Final Exam Project

## Foundations of Data Science: Programming and Linear Algebra (CDSCO1001U.LA_E22)

Raghava Mukkamala
rrm.digi@cbs.dk
Copenhagen Business School, Denmark

Deadline: {check your digital exam for exact date and time}

## Instructions

1. Please note that you have to upload the solutions before the deadline to the digital exam `http://exam.cbs.dk/`.
2. Please use Python 3 to answer the following questions as necessary.
3. Write a report explaining your assumptions/your design choices/ functionality etc. to understand your code in a better manner. It is also good practice to use comments extensively in your code so that it will be easy for other people to understand it. The report should conform to general formatting guidelines and academic standards that are expected for written projects at CBS.
4. You may answer the final exam project as a group project, and the number of people in the group is encouraged to be 2-4 as indicated in the course description: `https://kursuskatalog.cbs.dk/2022-2023/KAN-CDSCO1001U.aspx`. However, you can also work and submit your exam individually.
5. Finally, in case of Python code is involved in answering questions, you can submit a *jupyter* notebook containing all the code and report as markup in one file. Alternatively, you can also submit your code as one single python file (with .py extension) containing all the source code from different classes/modules/functions, etc.
6. Altogether, you need to submit the following files.
   (a) Report in Pdf format
   (b) A zip file containing i) Jupyter notebooks or exported Python code file (or else if you are using any other GUI like Pycharm, copy all the code into a single Python code file containing all the code).

> **Information** There are 3 questions in this assignment and you are expected to answer all the questions. Please feel free to use whatever format necessary [Latex, Word, plain paper, and pen] as you want, but make sure that it is clearly readable.

Question 1

**[Sub-Numpy]**

Previously and also as part of forthcoming courses, we will use a popular library for Machine learning called NumPy. The NumPy library allows a user to specify an array, create arrays of values 0s and 1s, provide implementations for the dot product, matrix multiplication, and much more. As part of this question, we will create our own implementation of a few functionalities supported by the NumPy Library. We will call our implementation SNumPy (for Sub-NumPy). SNumPy will be the name of the class you implement, and we will refer to it by the shorthand “snp” from here on. You are asked to provide an implementation in Python for the following few selected functionalities:

1. snp.ones(Int): the ones function takes an int parameter and returns an array (list) of length int parameter, and the array contains only ones. Example: snp.ones(5) = [1,1,1,1,1]

2. snp.zeros(Int): similar to the ones function, expect returns an array of zeros instead of ones.

3. snp.reshape(array, (row, column)): takes an array and converts it into the dimensions specified by the tuple (row, column). Hence this function converts from a vector to a matrix. For an example on reshape functionality of *numpy*, refer to fig. 1.

4. snp.shape(array) : returns a tuple with the matrix/vector’s dimension e.g. (# rows, # columns).

5. snp.append(array1, array2) : returns a new vector/matrix that is the combination of the two input vectors/matrices. Note that you can’t append a vector to a matrix and vice versa and therefore use suitable exception handling and throw/return user friendly error messages.

6. snp.get(array, (row, column)): returns the value specified by the coordinate point (row, column) of the array provided (can be vector or matrix).

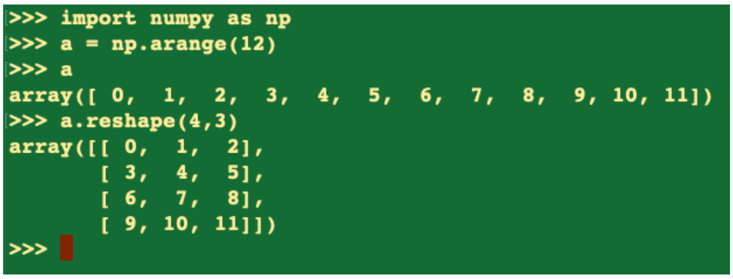
7. snp.add(array1, array1) : addition on vectors/matrices.

8. snp.subtract(array1, array1) : subtraction on vectors/matrices.

9. snp.dotproduct(array1, array1): computes the dot product between two arrays (which could be vectors or/and matrices) and returns an appropriate value. Use appropriate exception handling to output user-friendly error messages if the dot product cannot be performed between the given arrays.

10. **[Optional]** If you are not challenged enough by the above questions, then implement a solver for a system of linear equations using Gaussian elimination and row reduction rules for the functionality as depicted in `https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.solve.html`.

Please use proper exception handling to throw errors when input data is not in the right format (e.g., if you try to add a vector to a matrix or if the dimensions of the matrices don’t tally and so on).



```
>>> import numpy as np
>>> a = np.arange(12)
>>> a
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
>>> a.reshape(4,3)
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
>>> 
```

Figure 1: reshape functionality of *numpy*

**Additional Information**

- Additionally, you might have seen vectors have been represented different formats in the lectures and exercises to represent the fact that people will use different formats in expressing the vectors such as: $[\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}]$ OR $\begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \\ \boldsymbol{v_3} \end{bmatrix}$. So choose one format and use it all the way through the assignment. If you need to use special case vectors such as *column-vector* or *row-vector*, specify your format appropriately so that others can understand the notation.
- A vector can be thought of as a matrix with a single column or a single row, depending upon which notation you follow. For example, a vector [1,2,3,4] has the shape (1, 4), 1 row, and 4 columns. We are still using zero indexes, hence to get 4 from the above example array, we have to call the function: snp.get(array, (3,1)). Note the difference in shape and indexing.
- Our implementation will differ significantly from NumPy. Hence you can use NumPy as a reference on how to proceed, but don't build your solution on NumPy.

Question 2

**[Hamming's Code]**

As part of this question, you are going to use linear error-correcting codes invented by Richard W. Hamming in 1950 to detect [1, p.211], [2]. Hamming invented these linear error-correcting codes to detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors [3]. The linear error-correcting code that encodes four bits of data into seven bits by adding three parity bits. Hamming's (7,4) algorithm [3] can either correct any single-bit error, or detect all single-bit and two-bit errors as further described in [3]. Error-correcting codes are widely adopted in many kinds of transmission (including WiFi, cell phones, communication with satellites and spacecraft, and digital television) and storage (RAM, disk drives, flash memory, CDs,etc.) [1, p.211].

Hamming discovered a code in which a four-bit message is transformed into a seven-bit codeword. The generator matrix ($\mathbf{G}$), parity-check matrix ($\mathbf{H}$) discovered by Hamming is shown in fig. 2 and the Hamming's Decoder Matrix ($\mathbf{R}$) as shown in fig. 3. An encoding of a 4-bit binary value (word) $w$ is a 7-bit vector i.e. the codeword resulting from a matrix-vector product $c_w = \mathbf{G} * w$ [3].

1. Write a simple Hamming encoder program in Python, which, when given a 4-bit binary value, returns the resulting 7-bit binary vector codeword. Also implement the parity check functionality to see if there are any errors, that is to check whether $\mathbf{H} * c_w = \vec{0}$ holds, where $\vec{0}$ is zero vector.
2. Create a decoder program in Python, which, when given a 7-bit vector codeword, returns the original 4-bit vector word. That is, if we are given a 4-bit word ($w$), and we apply our encoder to return a codeword ($c_w = \mathbf{G} * w$), and then we apply the decoder matrix ($\mathbf{R}$) (fig. 3) to $c_w$, then it should return the original word, such that ($\mathbf{R} * c_w = w$).
3. Test your code by creating a few 4-bit vectors and running encode and then decode to check if you end up with the original 4-bit vector. Also, test your code with some errors and see if the parity check can identify the errors if so, to what extent.

**Hint** For more information on Hamming's Code, refer to the discussion at ch [4.7.3 - 4.7.5] from the book: Coding the Matrix. It helps to manually decode a value, such as [0,1,1,1,1,0,0], and form an intuition behind the decoding process.

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Figure 2: Hamming's Generator Matrix ($\mathbf{G}$) and Parity-check Matrix ($\mathbf{H}$) [3]

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3: Hamming's Decoder Matrix ($\mathbf{R}$) [3]

Question 3

**[Text Document Similarity]**

Create a python program that will compute the text document similarity between different documents. Your implementation will take a list of documents as an input text corpus, and it will compute a dictionary of words for the given corpus. Later, when a new document (i.e, search document) is provided, your implementation should provide a list of documents that are similar to the given search document, in descending order of their similarity with the search document.

For computing similarity between any two documents in our question, you can use the following distance measures (optionally, you can also use any other measure as well).

1. dot product between the two vectors
2. distance norm (or Euclidean distance) between two vectors .e.g. $|| \boldsymbol{u} - \boldsymbol{v} ||$

As part of answering the question, you can also compare and comment on which of the two methods (or any other measure if you have used some other measure) will perform better and what are the reasons for it.

ℹ

> **Hint** A text document can be represented as a word vector against a given dictionary of words. So first, compute the dictionary of words for a given text corpus containing the unique words from the documents of the given corpus. Then transform every text document of the given corpus into vector form, i.e., creating a word vector where 0 indicates the word is not in the document, and 1 indicates that the word is present in the given document. In our question, a text document is just represented as a string, so the text corpus is nothing but a list of strings.

# References

[1] P. N. Klein, *Coding the matrix: Linear algebra through applications to computer science*. Newtonian Press, 2013.

[2] Wikipedia, "Hamming code." `https://en.wikipedia.org/wiki/Hamming_code`, 11 2019.

[3] W. Article, "Hamming(7,4)." https://en.wikipedia.org/wiki/Hamming(7,4), 11 2019.