Homework 1: Computing the Distance Matrix and the Covariance Matrix of Data (Python Only)

Originally designed by Jane Doe

Given a set of N data, each with D entries, organized in the form of an $N \times D$ matrix,

$$X = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,D-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,D-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-1,0} & x_{N-1,1} & \cdots & x_{N-1,D-1} \end{bmatrix}$$

Each row of X is a horizontal vector $\overrightarrow{x_i}^T$, in which $\overrightarrow{x_i}$ is $D \times 1$.

Problem 0 (0 points):

Go over ComputeMatrices.py. Familiarize yourself with python and numpy. Here is a good starting point:
https://numpy.org/learn/

Problem 1 (10 points):

The goal is to compute an $N \times N$ matrix, whose entries are the pairwise Euclidean distance between any two data.

$$Z = \begin{pmatrix} \|\overrightarrow{x_0} - \overrightarrow{x_0}\| & \|\overrightarrow{x_0} - \overrightarrow{x_1}\| & \cdots & \|\overrightarrow{x_0} - \overrightarrow{x_{N-1}}\| \\ \|\overrightarrow{x_1} - \overrightarrow{x_0}\| & \|\overrightarrow{x_1} - \overrightarrow{x_1}\| & \cdots & \|\overrightarrow{x_1} - \overrightarrow{x_{N-1}}\| \\ \vdots & \vdots & \ddots & \vdots \\ \|\overrightarrow{x_{N-1}} - \overrightarrow{x_0}\| & \|\overrightarrow{x_{N-1}} - \overrightarrow{x_1}\| & \cdots & \|\overrightarrow{x_{N-1}} - \overrightarrow{x_{N-1}}\| \end{pmatrix}$$

You need to implement two functions:

1. The first one, uses a two level nested loop, iterating through all pairs (i, j) and compute the corresponding entry $Z_{i,j}$.
2. The second one, compute Z without any loop, but use **basic matrix operations** by numpy. This is vectorization.

Generate random matrices and profile the performance in terms of execution time, following the given example of entropy computation (EntropyComputation.py). The parameter for the x-axis should be N, which is also the number of columns of the Z matrix for this problem.

Note that there are ways to avoid loops in python without using matrix operation, e.g., the Einstein sum function `numpy.einsum`. But they are **NOT** what I want! If you profile it, you will see these methods are not much faster than loops. (You simply delegate the loops to python.)

Instead, you need to **vectorize** everything and use numpy's basic matrix operations, such as `add, multiply, matmul, inverse, diag, transpose, sqrt, eye, and ones`, etc. The key is that these methods are implemented in C and precompiled. So the loops are executed inside precompiled code. See the following links for references:

`http://codereview.stackexchange.com/questions/38580/fastest-way-to-iterate-over-`
`http://www.jesshamrick.com/2012/04/29/the-demise-of-for-loops/`
`http://cs231n.github.io/python-numpy-tutorial/#numpy-array-indexing`

Hint:

$$\|\overrightarrow{x_i} - \overrightarrow{x_j}\| = \sqrt{(\overrightarrow{x_i} - \overrightarrow{x_j})^T(\overrightarrow{x_i} - \overrightarrow{x_j})} = \sqrt{\|\overrightarrow{x_i}\|^2 - 2\overrightarrow{x_i}^T\overrightarrow{x_j} + \|\overrightarrow{x_j}\|^2}$$

Think about how to compute each of the three entries (for all (i, j) pairs) from the data matrix X.

Problem 2 (10 points):

Compute the correlation matrix ($D \times D$). The sample mean is $\overrightarrow{\mu} = \frac{1}{N}\sum_{i=0}^{N-1}\overrightarrow{x_i}$. It is a column vector with D entries: $\mu_0, \ldots, \mu_{D-1}$. The sample covariance matrix is

$$S = \begin{pmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,D-1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,D-1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{D-1,0} & s_{D-1,1} & \cdots & s_{D-1,D-1} \end{pmatrix} \text{ where } s_{i,j} = \frac{1}{N-1}\sum_{n=0}^{N-1}(X_{n,i}-\mu_i)(X_{n,j}-\mu_j)$$

Here $s_{i,i} = \frac{1}{N-1}\sum_{n=0}^{N-1}(X_{n,i} - \mu_i)^2$ is the sample variance of the i's dimension. The squareroot $\sqrt{s_{i,i}}$ is the i'th standard deviation, denoted by σ_i. Finally, the correlation matrix is

$$R = \begin{pmatrix} \frac{s_{0,0}}{\sigma_0\sigma_0} & \frac{s_{0,1}}{\sigma_0\sigma_1} & \cdots & \frac{s_{0,D-1}}{\sigma_0\sigma_{D-1}} \\ \frac{s_{1,0}}{\sigma_1\sigma_0} & \frac{s_{1,1}}{\sigma_1\sigma_1} & \cdots & \frac{s_{1,D-1}}{\sigma_1\sigma_{D-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{s_{D-1,0}}{\sigma_{D-1}\sigma_0} & \frac{s_{D-1,1}}{\sigma_{D-1}\sigma_1} & \cdots & \frac{s_{D-1,D-1}}{\sigma_{D-1}\sigma_{D-1}} \end{pmatrix}$$

You need to implement two functions:

1. Compute the correlation matrix using nested loops.

2. The second one, compute R without any loop, but use standard matrix operations by numpy (vectorization).

Generate random matrices and profile the performance. Following the given example of entropy computation. The parameter for the x-axis should be N, which is NOT the number of columns of the R matrix for this problem.

Again, for the second function, avoid loops, list comprehension, etc. Only use matrix operations (numpy).

Hint: first compute S, then a matrix of the denominators. To compute S efficiently, think about the row/col view of matrix multiplication.

Problem 3 (10 points):

Compute the pairwise distance matrix and correlation matrix for the three datasets from sklearn.datasets:

- Iris ($N = 150$, $D = 4$);
- Breast cancer ($N = 569$, $D = 30$);
- Digits ($N = 1797$, $D = 64$).

For each dataset, report the computational times with nested loops and without loop.

For distance matrix, there are 3×2 numbers to report. Put them in a table.

For correlation matrix, there are 3×2 numbers to report. Put them in a table (or try a bar-chart).

To Submit:

Code your work in python (Python 3). You are encouraged to code your work with Jupyter Notebook. For a quick introduction on Jupyter Notebook, see JupyterNotebookInstallation.pdf, SampleNotebook.ipynb, and https://realpython.com/jupyternotebook-introduction/

You are supposed to submit both the well-documented python files (.py) and the report. In the report, the following sections are required:

1. **Solutions:** For Problem 1 and Problem 2, description of your algorithms (the loop version and the vectorization version). The description should be independent of the programming language (so this is not an explanation of your code). Clearly indicate the dimensions of matrices in your work.

2. **Experiments:** For all three problmes, the timing plot/table and the discussion of the results. What is the control variable in this experiment? What did you observe? What do you learn from this observation?

Put all files together and pack into a single .zip file (NOT .rar or .tar.gz) for submission. It is preferred that the file is named ”LastNames-HW1.zip”. Include a readme, explaining which problem(s) you have finished,so I know how to grade. Content in the readme file:

1. What problems did you finish?
2. What platform did you use (linux? Mac? windows?)
3. Resources that helped me.

Only one submission per team, and full names of the members should be included in the Comments box of the submission page on Blackboard.