MLIA Fall 2017 Homewor 1: Computing the Distance Matrix and the Covariance Matrix of Data (Python Only)

Jane Doe

September 20, 2017

Due on October 10th. Late submission has no credits.

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 1 (2 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 matrix operations by numpy.

Generate random matrices and profile the performance. Following the given example of entropy computation.

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 operations such as `numpy.dot, numpy.multiply, numpy.divide`, 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 (2 points):

Compute the correlation matrix ($D \times D$). The sample mean is $\overrightarrow{\mu} = \frac{1}{N}\sum_{i=0}^{N}\overrightarrow{x_i}$. 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 matrix operations by numpy.

Generate random matrices and profile the performance. Following the given example of entropy computation.

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 (2 points): Compute the pairwise distance matrix and correlation matrix for the three datasets from sklearn:

- 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 (or try a bar-chart). This is one point.

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

To Submit:

You are supposed to submit both the python files (.py), modified from the given example, and the report. In the report, the following sections are required:

1. **The problem:** what were you asked to do (repeat my question in your own language). Why is it important?
2. **Solution:** what did you implement (the loop version and the matrix operation version). You need to explain the idea and show that it is correct. Finally, include the code in the document (only the function, not the whole script).
3. **Experiments:**
    - **Setting:** how do you evaluate the two functions? what are the different settings? how did you generate data? how did you profile the result? what did you measure (time in this case, in the future, maybe some accuracy etc)?
    - **Result and discussion:** Show the plot (performance comparison). Discuss the result: what did you observe? what do you learn from this observation?

Even if you do not know how to write paragraphs, try to start by answering my questions one by one.

Submit a zipped file and a readme

Put all files together and submit a zipped file. Include a readme, explaining which problem(s) have you finished. So I know how to grade. Content in the readme file:

1. What problems did you finish?
2. What python version (2.7? 3.6?)
3. What platform did you use (linux? Mac? windows?)