IE531: Algorithms for Data Analytics
Spring, 2020
End-Term Programming Assignment: Edo Liberty's Matrix Sketching Algorithm
Due Date: May 12, 2020

For this assignment you are going to implement Edo Liberty's Matrix Sketching algorithm [1]. Although this procedure was inspired by the family of *Misra-Gries Streaming Algorithms*, you do not have to implement it as such. **That is, a non-streaming version should be fine**.

You should take a look at Dr. Liberty's talk (click here) at the Simons Institute to get at a possible implementation-strategy for this assignment. Essentially we have a (large) data-matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$, where $n \gg m$[1].

You can use C++ or Python for the End-Term Programming Assignment. You have been exposed to the SVD-computations in Python already. If you are going to submit C++ code, you can use the *SVD-Decomposition* algorithm – SVD(A, D, U, V); – where A = U × D × V.t() in the *NEWMAT library*. As you can gather from the on line reference, NEWMAT assumes A is an $m \times n$ matrix where $m \geq n$. If $n > m$, then you have to work with A.t(), and make appropriate changes to U and V to make things work.

I have provided a hint in hint.cpp on Compass for the C++ users. Keep in mind that I tend to have my datasets appear in columnar-format. That said, you have to figure out the implementation details yourself. I have also provided two datasets on Compass. Sample outputs (using these two datasets) are shown in figure 1.

You have to take a measured-and-slow approach to getting this assignment done. The first-part involves understanding Dr. Liberty's algorithm. Following this, you have to think about a slightly-different implementation (from what is in his talk) – and this has to do with zeroing-out just the lowest-singular value (as opposed to everything below the median-singular-value). Then, for the C++ users, there is the issue with getting NEWMAT's SVD to do the needful. In the midst of all this, there are routine-things like computing the Frobenius Norm of a matrix (which BTW, can also be computed using its *Singular Values*). As you can see from the Frobenius Norms of the Original- and Sketch-Matrices in figure 1, my implementation comfortably meets the ϵ-specification (i.e. the reduction is Frobenius-Norm is not much at all).

[1] At least, as I like to think about it. Keep in mind that some authors may have more rows in the data-matrix than columns (i.e. $m \gg n$), you need to get used to these things! Nothing that "transposing of matrices" cannot fix! Liberty's paper uses the row-version, while his talk uses the column-version of his algorithm.

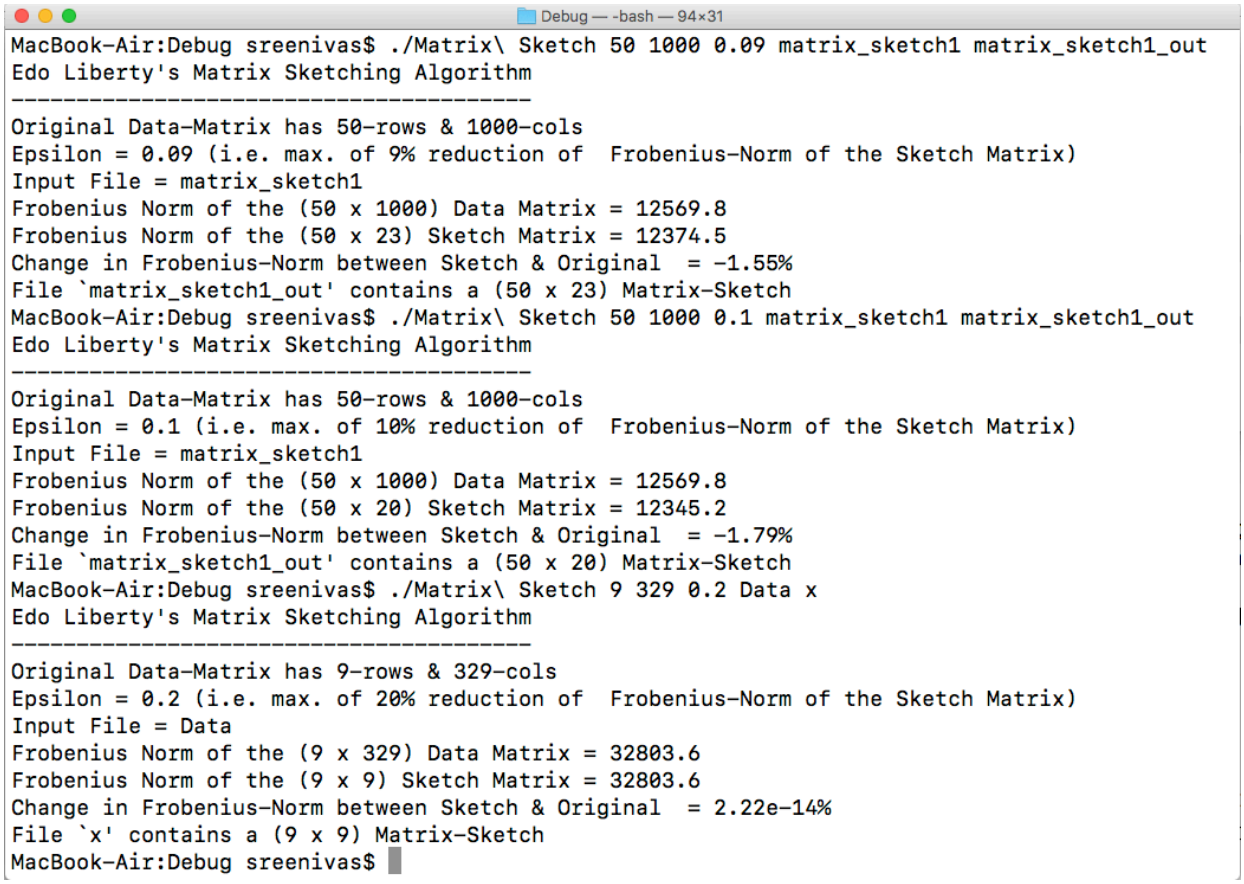

Figure 1: An illustration of the command-line variables.

References

[1] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 581–588, New York, NY, USA, 2013. ACM.