Student Name: Minh Phan
Course: AMATH 352
Email address: minhphan@uw.edu

**Homework 8**

---

# 1 Introduction and overview of the problem

Image processing is one of the important terms used in computer science, refers to the set of computational techniques for analyzing, compressing, enhancing and reconstructing images [5]. In the scope of this homework, we focus on the compressing and reconstructing aspects of image processing, which can be accomplished using extensive mathematical processes. Image compression and decompression technologies are essential in ensuring cost-effective solutions and feasibility in delivering images between parties [1].

# 2 Theoretical background and description of algorithm

Singular Value Decomposition, or SVD, is a concept in linear algebra referring to a factorization of a matrix $M$ of size $m \times n$ into the the form:

$$M = U\Sigma V^T$$

where:

- $U$ is an $m \times m$ orthogonal matrix,
- $\Sigma$ is an $m \times n$ diagonal matrix, and
- $V$ is an $n \times n$ orthogonal matrix

To understand how the algorithm for Singular Value Decomposition, define matrix V as:

$$V = [\underline{v}_1^T | \underline{v}_2^T | \cdots | \underline{v}_n^T]$$

as a matrix consisting of $n$ vectors that form the basis for $\mathbb{R}^n$. For a random vector $\underline{x}$, we can decompose this vector into scalars times basis vectors

$$\underline{x} = a_1\underline{v}_1 + a_2\underline{v}_2 + \cdot + a_n\underline{v}_n \rightarrow M\underline{x} = a_1M\underline{v}_1 + a_2M\underline{v}_2 + \cdot + a_nM\underline{v}_n$$

Apply applying the linear transformation M to each the basis vectors $\underline{v}_1, \underline{v}_2, ..., \underline{v}_n$, we have new vectors that can be decomposed by unit vectors (vectors that have length 1) $\underline{u}_1, \underline{u}_2, ..., \underline{u}_m$ times their respective magnitude $\sigma_1, \sigma_2, ..., \sigma_m$. Depending on the transformation we can manipulate the scalars into the magnitude values too, since they are scalars themselves.

$$(M\underline{v}_1, M\underline{v}_2, ..., M\underline{v}_n) \rightarrow (\underline{u}_1\sigma_1, \underline{u}_2\sigma_2, ..., \underline{u}_m\sigma_m)$$

Based on this representation of a vector $\underline{x}$, as we apply the linear transformation on all the basis vectors in matrix $V$, we can derive a different representation of the matrix product.

$$MV = U\Sigma \leftrightarrow MVV^T = U\Sigma V^T \leftrightarrow M = U\Sigma V^T$$

with $\Sigma$ being a diagonal matrix with entries as magnitudes corresponding to the unit vectors in $U$[6].
To better understand the report, I introduce some terms:

- Singular Value

  Singular values are the $\sigma_i$ entries in the matrix $\Sigma$ derived above. To explain this concept in simple terms, they refer to the "degree" of transformation, like how stretching or flattening the transformation is to the original basis[3].

- Rank r approximation and low rank approximation

  Low-rank approximation with rank r is a minimization problem where one recreates an approximated matrix subjecting to a rank constraint r[2]. A best r-rank approximation is given by zeroing out the trailing singular values after the $r^{th}$ one of the original matrix (whose singular values should be ordered) [4]. In other words, it is the process of representing the information in a matrix M using a matrix $\hat{M}$ that has a rank that is smaller than the original matrix [7].

# 3 Computational Results

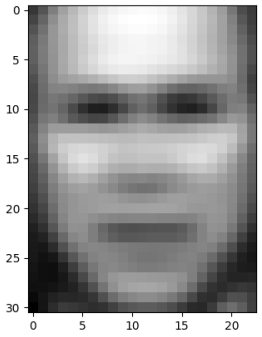

Figure 1: Image created from the mean of all rows of matrix A

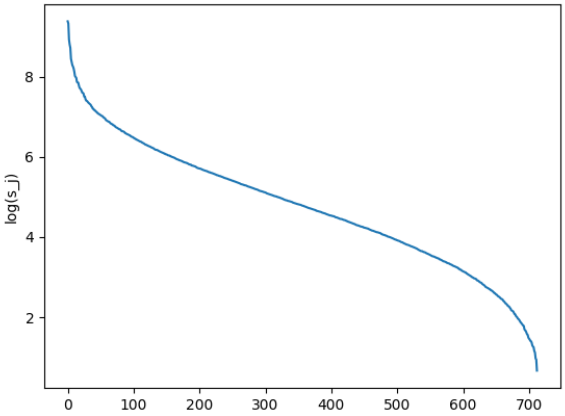


Figure 2: Plot of $j$ and $\log(\sigma_j)$

Comment on Figure 2: The plot has a consistently decreasing trend as Python internally ordered the singular values before outputting the array. The log function helps to emphasize this trend. As Python iterated through the array to graph, the singular values get closer to zero, hence appropriate for low-rank approximation (the closer the singular values are to 0, the less weight the corresponding left singular vectors have on the matrix approximation).

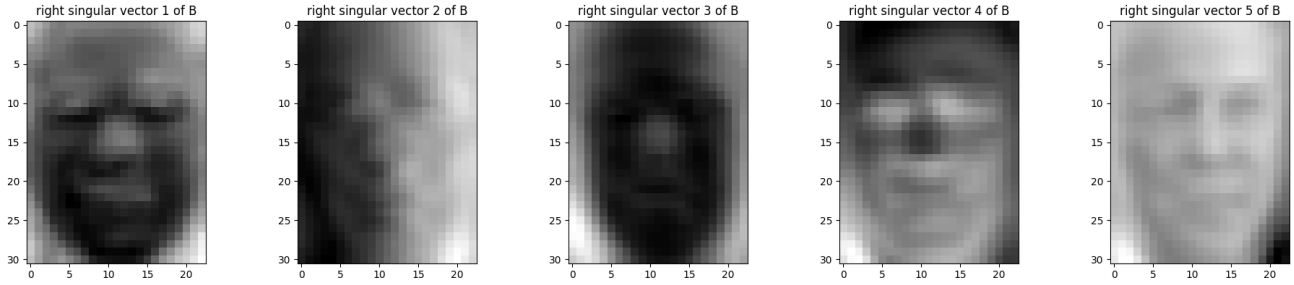


Figure 3: Images of the first 5 right singular vectors

Comment on Figure 3: The first 5 right singular vectors help to identify the discrete lighting areas on the image generated from the matrix B, and have different shaded areas that represent the most common patterns in all the pictures.

| Relative error (%) | 1 | 10 | 20 | 30 |
|---|---|---|---|---|
| Smallest value of r | 546 | 207 | 101 | 53 |

Table 1: The smallest value of rank r needed to achieve the relative error for approximation

On Table 1, we can see to achieve better approximations, we have to take account of very small-valued (close to zero) singular values at the end of the singular value array.


Figure 4: Images of the first 5 right singular vectors

Comment on Figure 4: The quality of the images increase as I decreased the relative error allowed. The rank of the matrices used in the approximation also has a direct impact on the approximation, as the larger the rank was the higher the quality of the approximation was and the more detailed it reflected the original picture.

# 4 Summary and Conclusions

Applying Singular Value Decomposition into recreating images through low rank approximation is a solid method for image compressing and decompressing. The amount of detail loss (relative error) is only 1% using the approximation method but one can save up to 20% of the data needed (considering the rank of the approximation over that of the original matrix). For the example above, even when one only uses 207 columns of the matrix (rank 207) – which is only 2/7 of the data needed – the amount of detail loss would only be 10%, which is very impressive and can save a lot of storage.

To generalize and conclude the subject at hand, we can see that no matter the type of information a random matrix $X$ encodes, it will have a matrix rank r, which is essentially the number of linearly independent columns (column rank) or rows (row rank) contained in the matrix. It's entirely possible (and common) for this matrix to have a rank that is smaller than the number of columns (or rows) in the matrix, and low-rank approximation can be used to "trim" away the redundant columns and reduce the size of the matrix but would still maintain an acceptable representation of the matrix itself [7]. This is even more useful in video compression where details between different frames are almost non-existent and low rank approximation is useful to point out the highlights between them, cutting off redundant data while compressing.

# References

[1] Ronald B. Arps and Thomas K. Truong. "Comparison of International Standards for Lossless Still Image Compression". In: *Readings in Multimedia Computing and Networking* (2002), pp. 113–123. DOI: 10.1016/b978-155860651-7/50095-4. (Visited on 11/25/2022).

[2] Wikipedia Contributors. *Low-rank approximation*. Wikipedia, Jan. 2021. URL: https://en.wikipedia.org/wiki/Low-rank_approximation.

[3] Wikipedia Contributors. *Singular value*. Wikipedia, Nov. 2022. URL: https://en.wikipedia.org/wiki/Singular_value (visited on 11/28/2022).

[4] Laurent El Ghaoui. *Low-rank approximation of a matrix*. inst.eecs.berkeley.edu. URL: https://inst.eecs.berkeley.edu/~ee127/sp21/livebook/l_svd_low_rank.html.

[5] The Editors of Encyclopaedia Britannica. *Image processing — computer science*. Encyclopedia Britannica. URL: https://www.britannica.com/technology/image-processing.

[6] Gregory W. Gundersen. *Singular Value Decomposition as Simply as Possible*. gregorygundersen.com, Dec. 2018. URL: https://gregorygundersen.com/blog/2018/12/10/svd/.

[7] Dustin Stansbury. *SVD and Data Compression Using Low-rank Matrix Approximation*. The Clever Machine, Aug. 2020. URL: https://dustinstansbury.github.io/theclevermachine/svd-data-compression#fnref:2 (visited on 11/28/2022).