Project 2: SVD applications

Lorenzo Vigo

Numeric Linear Algebra: 10^{th} December 2021

1. Least Squares problem

Write a program to solve the LS problem using SVD. Compute the LS solution for the datasets datafile and datafile2.csv that were used in pr4: QR factorization and least square problems. Compare the results using SVD with those obtained from the QR solution of the LS problems.

The main execution workflow in this problem is very simple: we just need to generate (or load the data) and solve the problem. These two steps have been abstracted from the main execution, so that different settings may be tested.

In terms of dataset loading, it should be noted that in the first exercise we need to generate the matrix A for a given degree. A will follow this structure:

$$A = \begin{pmatrix} 1 & x_1 & \dots & x_1^{d-1} \\ 1 & x_2 & \dots & x_2^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \dots & x_N^{d-1} \end{pmatrix}$$

The best performing degree is 9. It should be noted that the tested degrees are from 2 to 10 and that the selecting criteria is keeping the degree with the least error in the solution (which is around 10.8455 for both QR and SVD solving methods).

The implementation of SVD solving technique is quite simple as Numpy already provides us with a pseudoinverse method, which we will use to get A^+. The QR implementation is a little bit messier because we intended to unify the lines of code for both full-rank and rank deficient cases. In practice, all index accesses done by $[:r]$ in the QR routine will retrieve the whole matrices/vectors if we are in a full rank case, but only what should be taken in the rank deficient one.

We will analyze now the results for both datasets. For the first one, SVD and QR get a 10.8455 error with a solution with norm 137.2033. In the case of the second dataset, we get a 1.1496 error with a solution which norm is around 4774736.

2. Graphics compression

The SVD factorization has the property of giving the best low rank approximation matrix with respect to the Frobenius and/or the 2-norm to a given matrix. State properly the previous statement and write down the corresponding proofs for the Frobenius norm and the 2-norm.

Let A be our mxn matrix and let's consider that SVD decomposes our matrix A in 3 matrices U, S, V. The diagonal elements of S will be noted by σ_i and will be named singular values as in the project statement. In that way, we can define the k rank approximation of A through SVD decomposition as the following.

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{1}$$

In this definition, u_i and v_i denote the i-th column of U and V respectively. With this notation, now we may follow the steps of the proof found in [1].

We need two lemmata in order to prove what is requested:

Lemma 0.1. *The rows of A_k are the projections of the rows of A onto the subspace V_k spanned by the first k singular vectors of A.*

Lemma 0.2. $||A - A_k||_2^2 = \sigma_{k+1}^2$

Proofs to both lemmata can be found in [1]. Using the first lemma, we will try to prove that:

Theorem 0.3. *For any matrix B of rank at most k: $||A - A_k||_F \leq ||A - B||_F$.*

Proof. Let B be a matrix such that minimizes $||A - B||_F$ among all rank k or less matrices and therefore $||A-B||_F < ||A-A_k||_F$. Let V be the space spanned by the rows of B, which dimension is at most k.

Since B minimizes the error in Frobenius norm, following the first lemma, the rows of B should be the projections of the corresponding rows of A onto V. Otherwise, there would be another matrix B' with said rows that would minimize the error even further.

Since B is the projection of the corresponding row of A, $||A - B||_F^2$ is the sum of squared distances of rows of A to V. Given that A_k minimizes the sum of squared distance of rows of A to any k-dimensional subspace, the theorem is proven. □

Theorem 0.4. *For any matrix B of rank at most k: $||A - A_k||_2 \leq ||A - B||_2$.*

Proof. If A is of rank k or less, $||A - A_k||_2 = 0$. We will assume from now on that A is of rank greater than k. Let B be a matrix of rank at most k that minimizes the error. Due to the second lemma, $||A - B||_2 < \sigma_{k+1}$.

The null space of B, $Null(B)$, defined as the set of vectors v such that $Bv = 0$ has dimension at least $n - k$.

Let $v_1, ..., v_{k+1}$ be the first $k + 1$ singular vectors of A. Due to dimensions, there must be at least one vector $z \neq 0$ in $Null(B) \cap Span(v_1, ..., v_{k+1})$. Now,

$$||A - B||_2^2 \geq |(A - B)z|^2$$

Since $z \in Null(B)$, $Bz = 0$. Then,

$$||A - B||_2^2 \geq |Az|^2$$

Due to the decomposition of A:

$$|Az|^2 = |\sum_{i=1}^{n} \sigma_i u_i v_i^T z|^2$$

As u_i vectors are orthogonal, it follows that:

$$|Az|^2 = \sum_{i=1}^{n} \sigma_i^2 (v_i^T z)^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z)^2$$

Due to the ordering of the singular values and since $z \in Span(v_1, ..., v_{k+1})$:

$$|Az|^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z)^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} (v_i^T z)^2 = \sigma_{k+1}^2$$

Therefore, we have reached to the following contradiction:

$$||A - B||_2^2 \geq \sigma_{k+1}^2$$

And then, our theorem is proven.

□

Use the previous results to obtain a lossy compressed graphic image from a .jpeg graphic file. A .jpeg graphic file can be read as a matrix using the function scipy.ndimage.imread(). Use SVD decomposition to create approximations of lower rank to the image. Compare different approximations. The function scipy.misc.imsave() can be useful to save the approximated graphic files as .jpeg. The code must generate different compressed files for a given graphic file. Hence, to organize the output files, the name of the compressed file must reflect the percentage of the Frobenius norm captured in each compressed file. Use different .jpeg images (of different sizes and having letters or pictures) and compare results.

We have implemented the compression algorithm for color images (with 3 channels). The compression is done in each channel independently. The computed error that is shown in the generated filenames is the average of the error in each one of the channels.

We will experiment with 3 different pictures with different features. First, the file eye.jpg weighs 49KB with a size of 676x837. This picture includes letters, so we will be able to check the effects of compression on text.

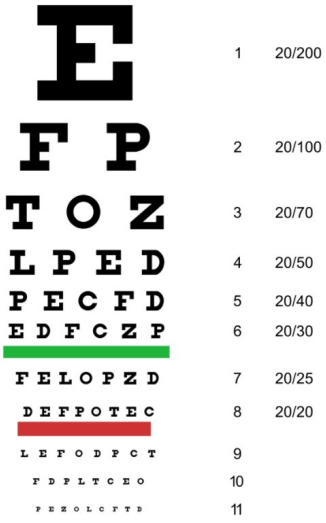


Figure 1: Original eye.jpg image used for experiments

We have tested several compression rates from 5 to 100, both included, with a step of 5. The rate represents the rate of information of the image that is kept after the compression. As you can see, the higher the rate, the lower the error.

As you can see in Figure 2, error is reduced drastically with rates around 25 and it tends towards 0 when getting closer to 100. The remaining error may come from errors in the decompositions carried out in the algorithm.

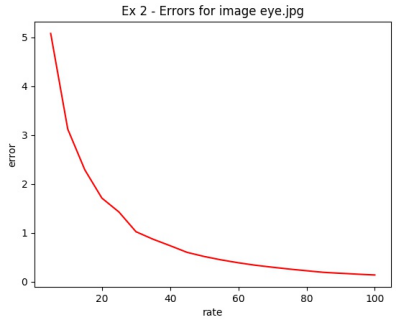
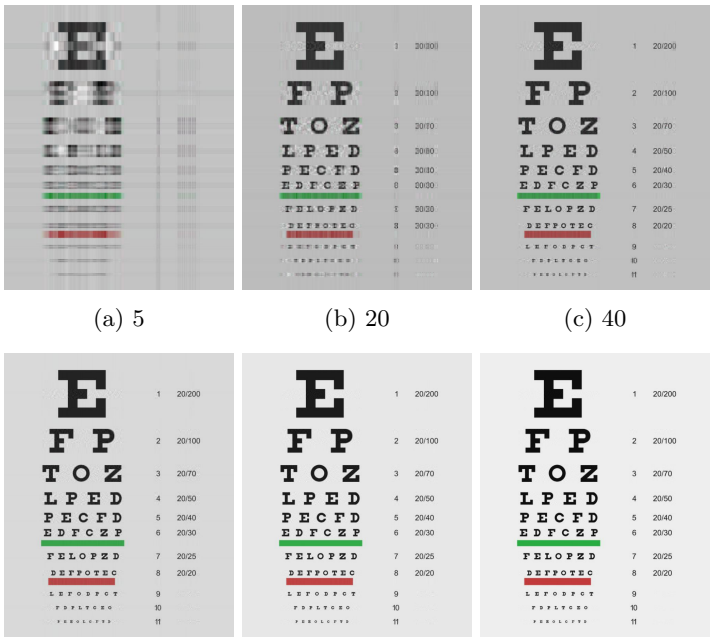


Figure 2: Error rates for eye.jpg image

Let us translate this to a more intuitive view, by showing the result images of some of the compression rates.



(a) 5 (b) 20 (c) 40

(d) 60 (e) 80 (f) 100

Figure 3: Compression of eye.jpg for several rates

With a 5 rate, the figures are indistinguishable and distorted. As we increase the conservation rate, letters start being more recognizable. The last two lines remain tricky at 40, but we would claim that everything is readable at 60.

We witness color loss especially in the white background. This phenomena persists even at 100 (but with a lower impact). This is due to white being rep-

resented by the color range extreme, which is harder to obtain precisely. We should note that black, being the other color range extreme, does not suffer that much thanks to the advantage of being represented by zeros.

Next experiments will be performed on a portrait. This image was extracted from thispersondoesnotexist.com in order not to use a real person's image for our project. The image weighs 427KB (much more than the first example) and it is as big as 1024x1024 pixels.

Figure 4: Original face.jpg image used for experiments

The behavior of the error is quite similar to the one seen in the experiments with text. Nonetheless, we could note that the initial error is higher and that the curve is steeper.

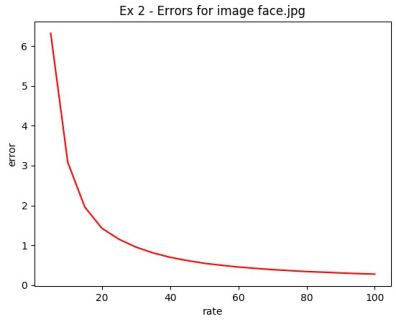
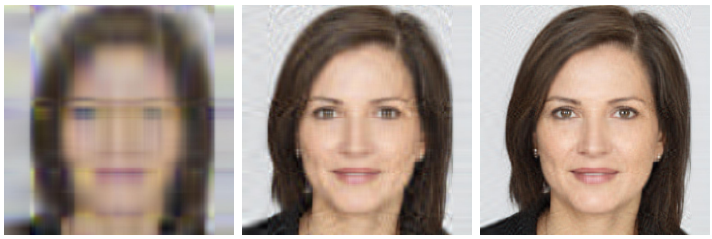


Figure 5: Error rates for face.jpg image

Let us visualize the compression as we did previously.



(a) 5 (b) 20 (c) 40

(d) 60 (e) 80 (f) 100

Figure 6: Compression of face.jpg for several rates

In the lowest rate, we can hardly recognize the facial features. However, from 20 conservation rate the face is recognizable. From there to higher rates, we experience an improvement in resolution and quality.

Our last experiment will be carried out on a landscape. The picture weighs 126KB and its size is 640x480px.

Figure 7: Original landscape.jpg image used for experiments

The error behavior is similar to the one seen in the previous experiments.

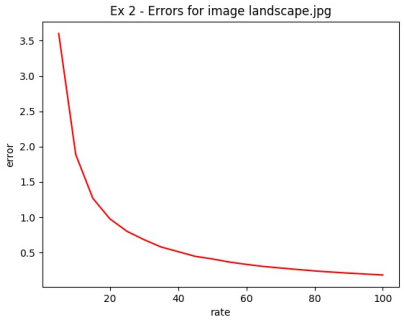
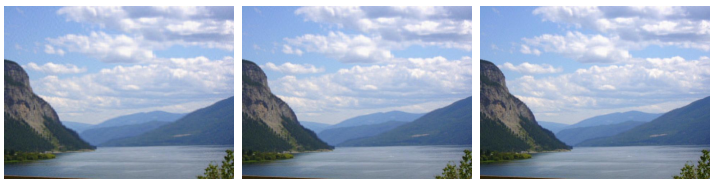


Figure 8: Error rates for landscape.jpg image

We will see now that is harder to get a good compression of this image, as it is full of little details that can not be preserved with high compression. It is not until close to a conservation rate of 100 that we can actually claim that the picture is not a painting.

(a) 5 (b) 20 (c) 40



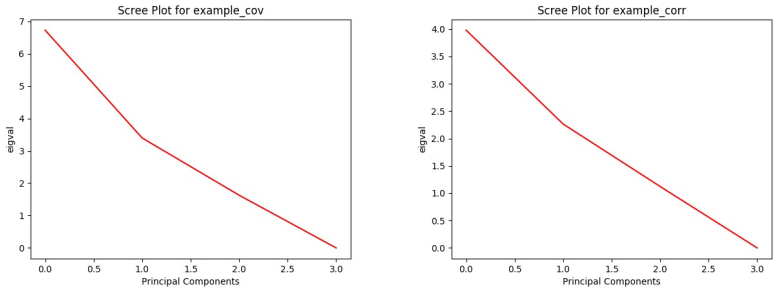
(d) 60 (e) 80 (f) 100

Figure 9: Compression of landscape.jpg for several rates

3. Principal component analysis (PCA)

In terms of implementation, we followed the usual abstract structure, in order to allow several running the same piece of code for different datasets and matrices. The construction of the correlation matrix and the output file generation got a little messy as I had to transpose matrices several times until they worked correctly.

As requested, we will try to explain the datasets with the proposed metrics.



(a) Covariance Matrix (b) Correlation Matrix

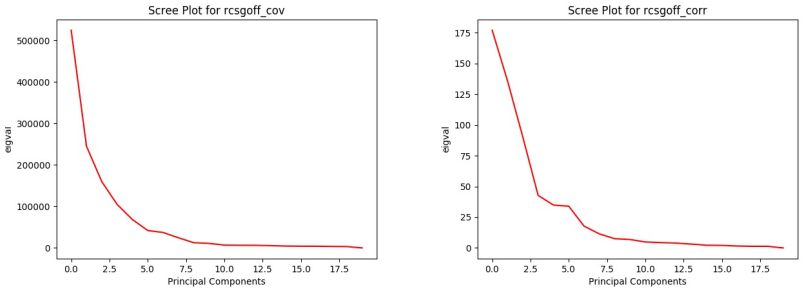
Figure 10: Scree Plots for example.dat

Scree plots show the eigenvalues of the pertinent matrix ordered by size. Those eigenvalues that lay on the left of the "elbow" of the curve are considered relevant, while the rest are considered redundant. The numbers of eigenvectors and eigenvalues in this dataset are low, but we could probably get rid of two of them for both matrices. That would mean there would be 2 principal components remaining.

We can use these plots to determine the Kaiser Rule too. Kaiser Rule consists in keeping only those principal components larger than 1. In this case, each matrix would be characterized by 3 principal components. Kaiser Rule generally tends to select more components than the "elbow" Scree plot method.

In terms of 3/4 rule, we need to find the minimum number of Principal Components needed to gather a 75% of accumulated variance among them. In this dataset, the Covariance Matrix presents a principal component with over 75% of the total variance on its own. Therefore, the 3/4 Rule on that matrix for this dataset gives us a total of 1 Principal Component. In the case of the Correlation Matrix, we need 2 Principal Components. This analysis has been checked through code.

Let's repeat the same analysis for the RCsGoff.csv dataset.



(a) Covariance Matrix (b) Correlation Matrix

Figure 11: Scree Plots for RCsGoff.csv

The elbow Scree Plot method in this case would demand us to keep 5 Principal Components in the case of the Covariance Matrix. We can appreciate two little elbows in the Correlation Matrix Scree Plot. We would say we should keep 6 or 7 Principal Components, taking into account the second elbow instead of the first one.

The Kaiser Rule on these two matrices detects 19 Principal Components linked to eigenvalues higher than 1.

Lastly, the 3/4 rule guides us to keep only 2 Principal Components for both matrices. This is due to the highest variance Principal Components accumulate up to 72% and 15% variance in the Covariance Matrix case and 50% and 30% in the case of Correlation Matrix.

References

[1] Avrim Blum, John Hopcroft, and Ravindran Kannan, *Foundations of Data Science*. Cornell University, 2018.