UNIVERSITÉ DE FRANCHE-COMTÉ

Using the SVD Method for Image Reconstruction

Master of Applied Mathematics with Statistical Modelling

Narain Ritish

A.V Hurtado Quiceno

Contents

1 Introduction

The singular value decomposition (SVD) factorization technique is used in data science and machine learning to analyze data. This method reduces the dimensions of a data in matrix form while preserving the most relevant characteristics and reduces noise in the data.

Singular Value Decomposition (SVD) is used in this project to restore grayscale images. A grayscale image carries only intensity information with values $[0, 1]$ representing a single sample of light. Grayscale images, also known as black and white or gray monochromes, consist entirely of shades of gray having a range of contrast depending on the intensity.

In computer imaging, grayscale images differ from one-bit bi-tonal black-and-white images having two colors only: black and white (also called bilevel or binary images) and the grayscale image has many shades of gray. First we will normalize our matrix in order to scale the values (subtracting the mean and dividing by the standard deviation).

In section 2, we go through an overview of the mathematical theory of the SVD method, its proof and some observations.

In section 3, we perform a scenario analysis using R Studio. We will show the analysis of the svd method by both scaling and without scaling the initial image matrix. We will compare the two approach with metrics of mean square error (MSE) and peak signal-to-noise ratio (PSNR) to compare the image compression quality. We will use a grayscale picture of Montmartre in Paris as our original picture.

Figure 1: Montmartre

2 Mathematics of the Singular Value Decomposition (SVD)

Providing a characterization of a matrix is the purpose of diagonalizing. We can determine the rank, eigenvalues and the invertibility of a matrix. For a squared matrix $A \in \mathcal{M}_{n\times n}(\mathbb{R})$, it is useful to find a decomposition $A = Q^{-1}DQ$, where Q is an invertible matrix and D diagonal but this decomposition does not always exists.

On the other hand, if A is not a square matrix, the Gauss elimination provides a factorization of type P invertible and Q diagonal. Furthermore, if A is real, we can choose S and T orthogonal real matrices and Σ diagonal to have the singular value decomposition (SVD).

This section provides some preliminary SVD results and definitions [Nic19],[SK19].

Definition 1 *Let $A \in \mathcal{M}_{m\times n}(\mathbb{R})$ then the image space and column space of A are defined as follows:*

1. *The image of A is defined by*

$$\text{Im}(A) = \{Ax; x \in \mathbb{R}^n\}. \tag{1}$$

2. *The column space of A is defined by*

$$\text{col}(A) = \text{span}\{A_i; A_i \textit{a column of } A\} \tag{2}$$

The two spaces coincides, i.e for $A \in \mathcal{M}_{m\times n}(\mathbb{R})$ we have that $\text{Im}(A) = \text{col}(A)$.

Lemma 1 *Let $A \in \mathcal{M}_{m\times n}(\mathbb{R})$. Then:*

1. *The matrices A^TA and AA^T have real and non-negative eigenvalues.*

2. *A^TA and AA^T have the same set of positive eigenvalues.*

Definition 2 *Let λ_i for $i = 1, 2, \ldots, n$ be the eigenvalues of $A^\intercal A$ (or $AA^\intercal$), then the real numbers $\sigma_i = \sqrt{\lambda_i}$ for $i = 1, 2, \ldots, n$ are called the singular values of the matrix A.*

Theorem 1 (Singular Value Decomposition) *Let $A \in \mathcal{M}_{n\times p}(\mathbb{R})$ of $\text{rank}(A) = r \leq \min(n, p)$, there exists:*

1. *A matrix $S \in \mathcal{M}_{n\times r}(\mathbb{R})$ be of $\text{rank}(A) = r$ and semi-orthogonal i.e $S^\intercal S = I_r$.*

2. *A matrix $\Sigma \in \mathcal{M}_{r\times r}(\mathbb{R})$ be of $\text{rank}(A) = r$ and diagonal i.e $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.*

3. *A matrix $T \in \mathcal{M}_{p\times r}(\mathbb{R})$ be of $\text{rank}(A) = r$ and semi-orthogonal i.e $T^\intercal T = I_r$.*

Algorithm 1 (SVD Algorithm) *Let $A \in \mathcal{M}_{n\times p}(\mathbb{R})$ of rank $r \leq \min(n, p)$. We have the matrices S, T and Σ such that $A = S\Sigma T^\intercal$ as follows:*

1. *Diagonalize the matrix $A^\intercal A$ in order to find the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ with corresponding (orthonormal) eigenvectors $t_1, t_2, \ldots, t_p$.*

2. *Reorder the t_i (if necessary) to guarantee that the nonzero eigenvalues are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$ and $\lambda_{r+1} = \cdots = \lambda_p = 0$.*

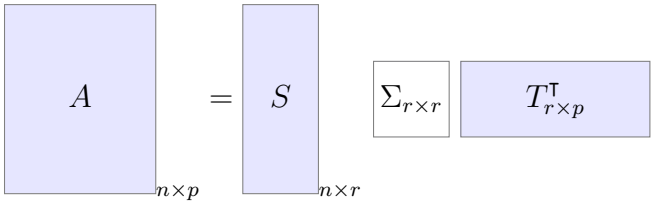
3. *The semi-orthogonal matrix T in the SVD is $\mathcal{T} = [\underbrace{t_1 \,|\, t_2 \,|\, \cdots \,|\, t_r}_{T} \,|\, \underbrace{t_{r+1} \,|\, \cdots \,|\, t_p}_{T_*}]$.*

4. *Define $s_i = \frac{1}{\|At_i\|} At_i$ for $i = 1, 2, \ldots, r$. $\{s_1, s_2, \ldots, s_r\}$ is orthonormal in $\mathbb{R}^n$ and we use this base to extend it to an orthonormal basis using the Gram-Schmidt method.*

5. *The $n \times n$ semi-orthogonal matrix S in the SVD method is $S = [s_1 \,|\, \cdots \,|\, s_r \,|\, \cdots \,|\, s_n]$.*

6. *The singular values for A are $\sigma_1, \sigma_2, \ldots, \sigma_p$ where $\sigma_i = \sqrt{\lambda_i}$ for each i. Hence the nonzero singular values are $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, and so the singular matrix of A in the SVD is $\Sigma = \text{diag}\,(\sigma_1, \ldots, \sigma_r)$.*

7. *Thus, $A = S\Sigma T^\intercal$ is the SVD decomposition for the matrix A,*



(3)

In 1936, Carl Eckart and Gale J. Young were the first to demonstrate the singular value decomposition for rectangular and complex matrices. An important property of the SVD is that it provides an optimal low-rank approximation to a matrix.

Theorem 2 (Eckart-Young) *The optimal rank-r approximation to* **A** *minimization, is given by the rank-r SVD truncation* $\tilde{\mathbf{A}}$ *:*

$$\underset{\mathbf{A},\, s.t.\ \text{rank}(\tilde{\mathbf{A}})=r}{\text{argmin}} \quad \|\mathbf{A} - \tilde{\mathbf{A}}\|_F = \tilde{\mathbf{S}}\tilde{\Sigma}\tilde{\mathbf{T}}^\intercal$$

Observation 1 *The SVD decomposition is not unique but singular values are unique. However the columns of S and T are only unique and the Gram-Schmidt provides an expla-*

nation for this. The order of eigenvectors arranged corresponding to the zero or repeated eigenvalues. For instance, if we consider the following matrix:

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{4}$$

The symmetric matrix $A^{\intercal}A$ has two eigenvalues $\lambda_1 = 3$ with multiplicity 2 and $\lambda_2 = 0$. Due to the way we write the two eigenvectors associated to the eigenvalue $\lambda_1 = 3$ as well as the Gram-Schmidt method, the matrix T and S are not unique.

Definition 3 *The fundamental subspaces of an $n \times p$ matrix A are:*

$$\text{row}A = \text{span}\{\mathbf{x} \mid \mathbf{x} \textit{ is a row of } A\}$$

$$\text{col}A = \text{span}\{\mathbf{x} \mid \mathbf{x} \textit{ is a column of } A\}$$

$$\ker A = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{0}\}$$

$$\ker A^T = \left\{\mathbf{x} \in \mathbb{R}^n \mid A^T\mathbf{x} = \mathbf{0}\right\}$$

Observation 2 *If $A = S\Sigma T^{\intercal}$ of any SVD for the real $n \times p$ matrix A, any orthonormal bases of S and T provide orthonormal bases for each of these fundamental subspaces.*

We are going to prove this, but first we need three properties related to the orthogonal complement $S^{\perp}$ of a subspace S of $\mathbb{R}^n$, where:

$$U^{\perp} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{u} \cdot \mathbf{x} = 0 \text{ for all } \mathbf{u} \in U\}$$

The orthogonal property plays an important role in the SVD theorem. For now we need:

Lemma 2 *If A is any matrix then:*

1. $(\text{row } A)^{\perp} = \text{null } A$ *and* $\text{col}(A)^{\perp} = \text{null } A^T$.

2. *If U is any subspace of $\mathbb{R}^n$ then $U^{\perp\perp} = U$.*

3. *Let $\{\mathbf{f}_1, \ldots, \mathbf{f}_m\}$ be an orthonormal basis of $\mathbb{R}^m$. If $U = \text{span}\{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$, then*

$$U^{\perp} = \text{span}\left\{\boldsymbol{f}_{k+1}, \ldots, \mathbf{f}_m\right\}$$

Theorem 3 *Let A be an $n \times p$ real matrix, let $A = S\Sigma T^{\intercal}$ be any SVD for A where S and T are semi-orthogonal of size $n \times n$ and $p \times p$ respectively, and let*

$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}_{n \times p} \quad \textit{where} \quad D = \text{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_r)\textit{, with each } \lambda_i > 0$$

We write $S = \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_r & \cdots & \mathbf{s}_n \end{bmatrix}$ and $T = \begin{bmatrix} \mathbf{t}_1 & \cdots & \boldsymbol{t}_r & \cdots & \mathbf{t}_p \end{bmatrix}$, so $\{\boldsymbol{s}_1, \ldots, \mathbf{s}_r, \ldots, \mathbf{s}_n\}$ and $\{\mathbf{t}_1, \ldots, \mathbf{t}_r, \ldots, \mathbf{t}_p\}$ are orthonormal bases of $\mathbb{R}^n$ and $\mathbb{R}^p$ respectively. Then:

1. *$r = \text{rank}\, A$, and the singular values of A are $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_r}$.*

2. *The fundamental spaces are described as follows:*

   a. *$\{\mathbf{s}_1, \ldots, \mathbf{s}_r\}$ is an orthonormal basis of $\text{col}\, A$.*

   b. *$\{\mathbf{s}_{r+1}, \ldots, \mathbf{s}_n\}$ is an orthonormal basis of null A^T.*

   c. *$\{\mathbf{t}_{r+1}, \ldots, \mathbf{t}_p\}$ is an orthonormal basis of A.*

   d. *$\{\mathbf{t}_1, \ldots, \mathbf{t}_r\}$ is an orthonormal basis of row A.*

Observation 3 *We can give an SVD decomposition where S and T are squared and orthogonal matrices. In this case the matrix Σ is rectangular with the values $\sigma_i > 0$ for all $i =, \cdots, r$ and the other columns are completed by the null eigenvalues. In this case the matrix Σ is given by $\Sigma = \Big(diag(\sigma_1, \cdots, \sigma_r) \mid 0\Big)$.*

3 Application of the SVD method with R

As we saw in the above section, the SVD method decompose a matrix A into $S\Sigma T^{\intercal}$ where S, T are semi-orthogonal matrices and Σ is a diagonal matrix. Using this method, we are able to extract the singular values to reconstruct our image. The SVD method is widely used in statistics and signal processing. In this project, we will study and applied the SVD method to compress an image. We will use the photo of Montmartre in Paris (Figure 1, Montmartre 1).

3.1 SVD Implementation

We use the function $readPNG$ in order to translate the image from a $.png$ and convert the pixel data into a matrix A of size $n \times p = 501 \times 750$. We also created a function called svd_image that is used to reconstruct the image using the svd decomposition as follows:

1. We use the function svd in R SVD decomposition of the matrix A as a first step to reconstruct the image.

2. We convert our image to gray scale intensity with the values $[0, 1]$ using the function $image$ from R.

3. We select the number of singular values that we want to keep and generate our image.

3.2 SVD process

As a first step of the process, we compare the reconstructed photo from the SVD matrix before and after standardization. We will normalize the matrix A by subtracting the mean μ and dividing by the standard deviation σ of A.

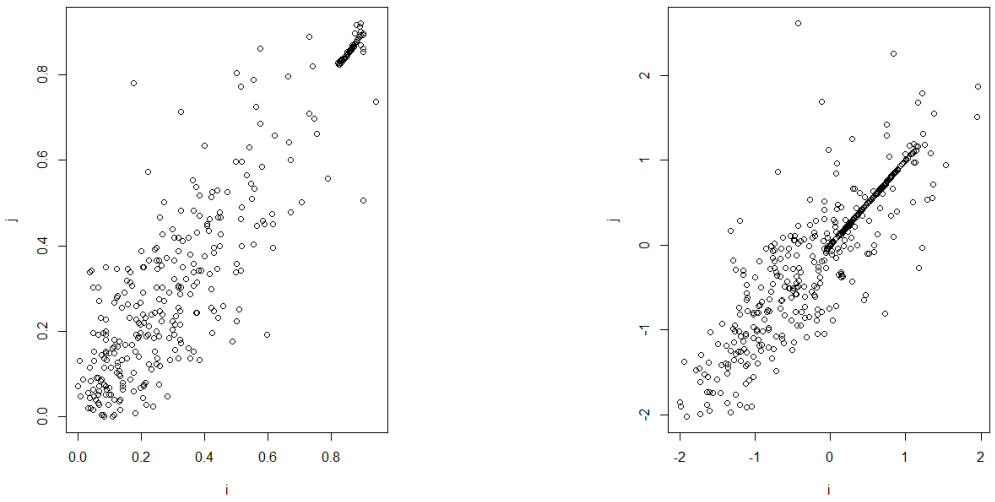


Figure 2: Before standardization (left) and after standardization (right) of the matrix A.

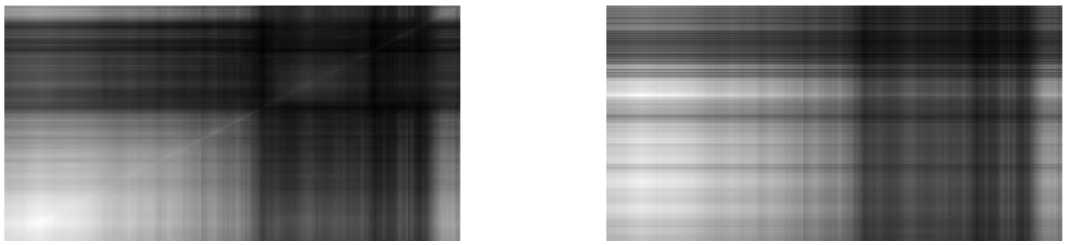
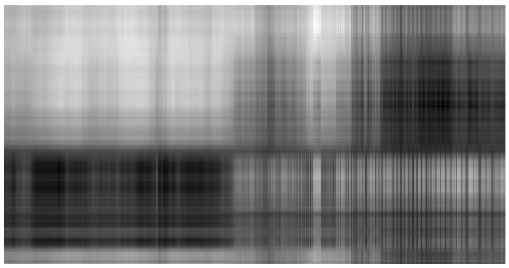


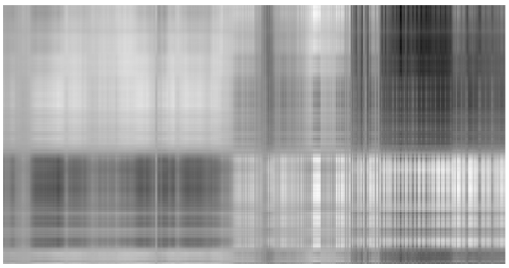
Figure 3: Correlation matrices $AA^\intercal$ (left) and $A^\intercal A$ (right) for a matrix A.

Figure 4: Original photo

Figure 5: Retaining 90 singular values (Standardized SVD matrix)



(a) 2 singular values



(b) 2 singular values

(c) 20 singular values

(d) 20 singular values

(e) 30 singular values

(f) 30 singular values

(g) 50 singular values

(h) 50 singular values

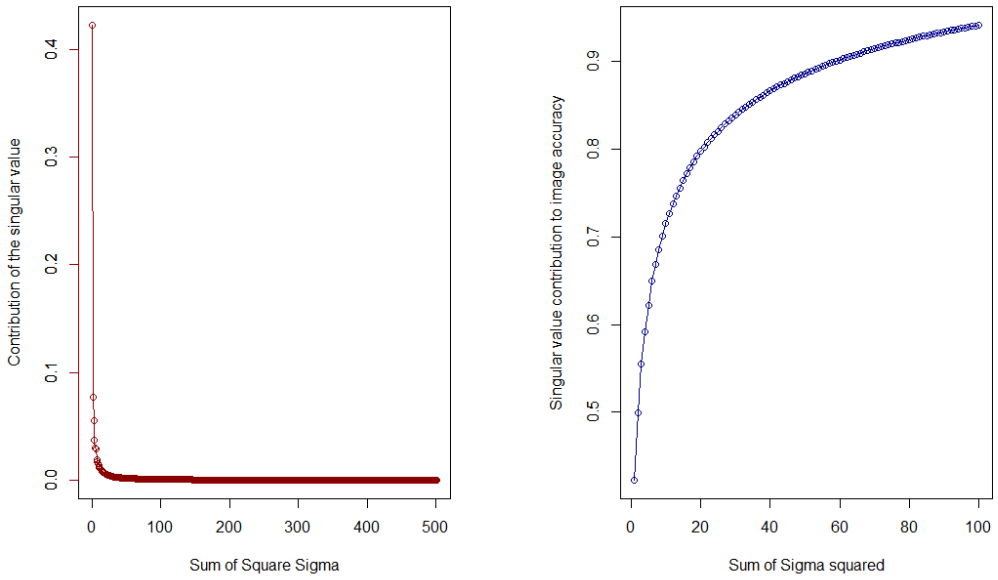
(i) 90 singular values

(j) 90 singular values

Figure 6: Non-Standardized (left) and Standardized (Right)

3.3 Scenario Analysis

3.3.1 The singular values



(a) (b)

Figure 7: Contribution of the singular values (left) and contribution to image accuracy (right)

Weight of one singular value σ_i 7a:

$$\frac{\sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}. \tag{5}$$

Represents the accuracy percentage of the image 7b:

$$\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}, \tag{6}$$

where r is the number of singular values ($rank(A)$).

As shown in figure 7a, we can see the first 20 singular values are the largest and contributes

to almost 80% of the image accuracy from figure 7b. We showed that we have a close representation of our image with 90 singular values.

3.3.2 Compression size of the Images

We show the different compression size of our images with different singular values. We will compare both the images before and after standardization in the table below 1.

Table 1: Compression size comparison of our images.

	NON-STANDARDIZED		*STANDARDIZED*		S	Σ	$T^\intercal$
# Singular Values	Compression (Kb)	Type	Compression (Kb)	Type	$n \times r$	$r \times r$	$r \times p$
2	116.2	8 *bit*	110.3	8 *bit*	501×2	2×2	2×750
20	159.8	8 *bit*	147.3	8 *bit*	501×20	20×20	20×750
30	161.3	8 *bit*	150.7	8 *bit*	501×30	30×30	30×750
50	168.4	8 *bit*	150	8 *bit*	501×50	50×50	50×750
90	170.4	8 *bit*	151.3	8 *bit*	501×90	90×90	90×750
ORIGINAL	257	8 *bit*	257	8 *bit*	501		750

From the table above 1, we showed that we have a better compressed image size when standardizing our image. Using the SVD, we get a more compressed image using less singular values and we used 90 singular value to get approximately 90% of our image resolution.

3.3.3 MSE and PSNR Metrics

The Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are the two most popular metrics used in image processing to compare the error and compression quality of an image.

The PSNR is expressed in terms of logarithmic decibel scale (dB) between the original photo and the image retaining i singular values that measures the quality of the images. Good image quality is considered when the PSNR values ranges between 30 and 50 dB for an 8-bit image [Wel99]. The PSNR is given by the following expression:

$$\text{PSNR} = 20 \log_{10} \left(\frac{\max A_{ij}}{\sqrt{\text{MSE}}} \right), \text{for } 1 \leq i \leq n, 1 \leq j \leq p \tag{7}$$

where A_{ij} are the values of the matrix A.

The MSE indicates the error between the compressed image and the original. The MSE decreases when we use more singular values and the PSNR increases which shows better signal strength.The MSE is given by the following expression:

$$\text{MSE} = \frac{1}{np} \sum_{i=0}^{n-1} \sum_{i=0}^{p-1} \left\| A_{i,j} - \tilde{A}_{i,j} \right\|^2 \tag{8}$$

where n is the number of rows of pixels, p is the number of columns of pixels and $\tilde{A}$ is the pixel matrix of our compressed image.

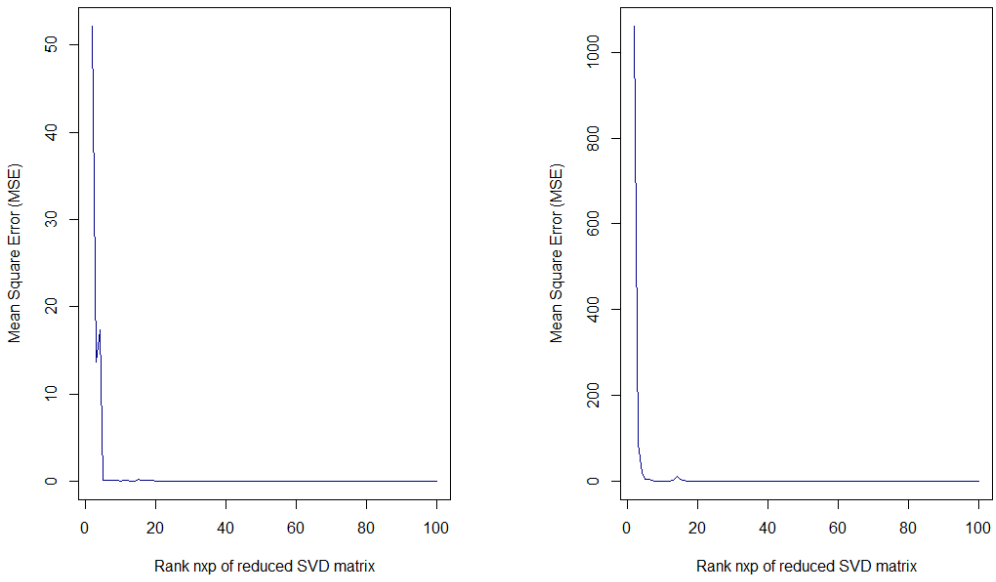


Figure 8: MSE of non-standardized (left) and standardized (right) matrix with different ranks.

Table 2: MSE and PSNR comparison for the standarized and non-standardized matrix.

	NON STANDARIZED		*STANDARDIZED*	
# Singular Values	MSE	PSNR	MSE	PSNR
2	52.19354	-18.86259	1061.395	-43.1052
20	0.03649668	12.97586	0.1636612	-3.29775
30	0.01782308	16.81761	0.0080761	14.81973
50	0.0003870735	34.12207	0.01645694	16.46402
90	0.0004566511	33.40415	1.017869e-08	78.77285

From the figure 9, we can see of having a better signal strength with standardization of the matrix and enables to capture the maximum information with each additional singular value to reconstruct the image. From the figure 9, we see of having higher MSE with standardization but it decreases drastically when using above 10 singular values. The table 2 shows we have an image with much better reconstruction quality with standardization with 78.77 db compared with 33.40 db without standardization with 90 singular values.

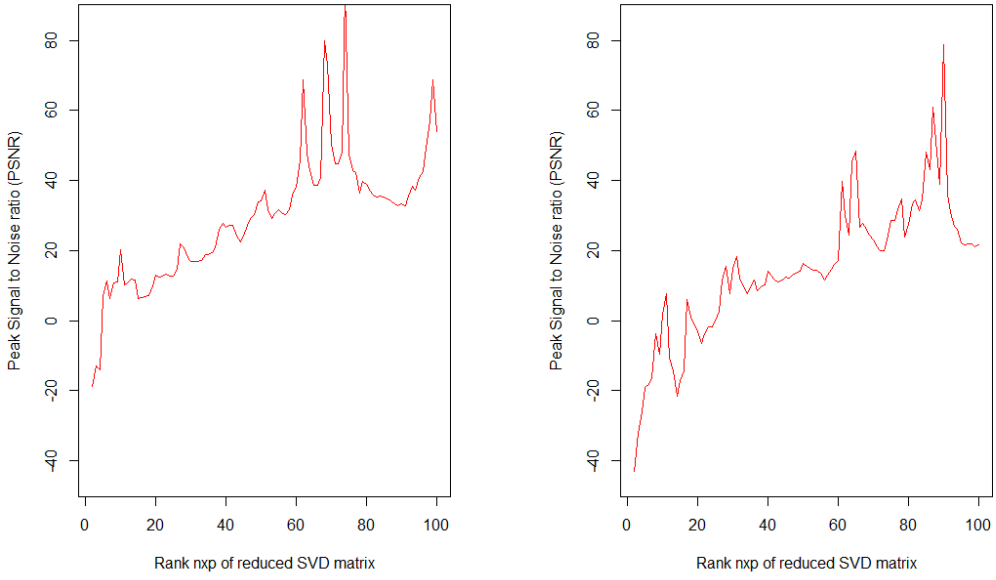


Figure 9: PSNR of non-standardized (left) and standardized (right) matrix with different ranks.

We note that the PSNR is not always better with standardization if we use 20 singular values for example. However, standardization lowers the variance of the signal range and capture maximum information.

3.3.4 The Condition Number

The condition number κ of a matrix indicate the stability of the equation $Ax = b$. It is ill-conditioned if a small perturbation in x changes the vector b by a large amount and gives a large κ value. Otherwise, the problem is said to be well conditioned and have a small value κ. We will observe the change of κ of our SVD matrix by using the singular eigenvalue to calculate κ as follows:

$$\kappa(A) = \frac{\sigma_{\text{max}}(A)}{\sigma_{\text{min}}(A)}, \tag{9}$$

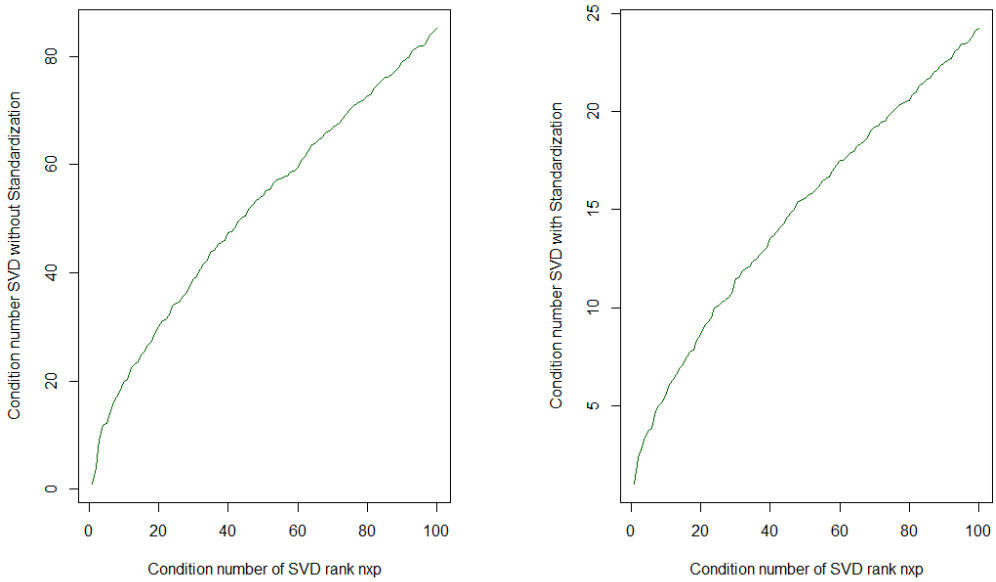


Figure 10: The condition number of non-standardized (left) and standardized (right).

We compare κ of the matrix $A = S\Sigma T^{\intercal}$ using the singular values of the Σ matrix before and after the standardization, having a rank r which is also the number of singular values.

Table 3: Condition number comparison

	NON STANDARIZED	STANDARDIZED		
# Singular Values	$\kappa(A)$	$\kappa(A)$	$\kappa^2(A)$	$\kappa(A^\intercal A)$
2	4.102	2.341	5.481	5.481
20	30.051	8.648	74.794	74.794
30	38.916	11.461	131.354	131.354
50	54.301	15.551	241.839	241.839
90	78.947	22.474	505.068	505.068

From the above table 3, we can see the κ values of the total singular values used to reconstruct our image and the stability of $Ax = b$. We have also confirmed the result from the equation below [Gan18] :

$$\kappa(A^\intercal A) = \kappa((S\Sigma T^\intercal)^\intercal S\Sigma T^\intercal) = \kappa(T\Sigma^T\Sigma T^\intercal) = \frac{\sigma_1^n}{\sigma_n^2} = \kappa(A)^2. \tag{10}$$

From our analysis, we can observe that the the solution x of $Ax = b$ becomes more inaccurate with more singular values used. However, we can observe a better conditioning when we standardize our matrix before the SVD decomposition.

3.4 Computational time

In this section, we are comparing the computation time (in seconds) by the SVD function in R with the matrix A before and after standardization of our image pixel matrix.

CPU time	Non Standardized	Standardized
user	0.28	0.30
system	0.02	0.00

From the table below, we show the R studio time taken and (user CPU time) and the system CPU time of the function svd_image2 and svd_image .1 before and after standardizing the matrix A.

Table 4: Computational time of the svd_image before and after standardization.

	Non Standardized		*Standardize*	
# Singular Values	user (CPU time)	system (CPU time)	user (CPU time)	system (CPU time)
2	0.16	0.55	0.14	0.57
20	0.24	0.58	0.30	0.54
30	0.28	0.61	0.28	0.61
50	0.21	0.64	0.21	0.65
90	0.25	0.61	0.27	0.61

From the table 4 above, we can observe a slight difference of the computational time taken to reconstruct the image in both standardized and non-standardized matrix. We can observe the noticeable difference of user time of 8 and 16 seconds when using the first 20 singular values and we capture the most information for our image reconstruction with those number of singular values.

4 Conclusion

From our study and analysis, we have demonstrated that we do not need all the singular values to reconstruct an image of a good quality. In our case, we have kept 90 to reconstruct our image and showed that the image is almost exactly the same with a much smaller matrix and less singular values.

Also, we have reconstructed our initial image by first scaling the values of the image pixel matrix with standardization to compare the svd without scaling the pixel values. Using the MSE and PNSR metrics for analysis, we showed the mean squared error between our original matrix and the image reconstructed using our SVD matrix decreases with more singular values and the PNSR going in the opposite direction showing better signal strength.

To conclude, we showed that scaling the values of our matrix gives us better signal strength that captures the most details with each additional singular used.

5 Bibliography

References

[Wel99] Stephen Welstead. “Fractal and wavelet image compression techniques”. In: SPIE Publication, 1999.

[Gan18] Walter Gander. “The Singular Value Decomposition”. In: ETH Zurich personal communications, 2018.

[Nic19] W.Keith Nicholson. “Linear Algebra with Applications”. In: (Jan. 2019).

[SK19] Brunton Steven and J Kutz. “Chapter 1: Singular Value Decomposition”. In: Apr. 2019, p. 3. ISBN: 9781108380690.

Annexes

.1 R Code

```
# Le Code de la décomposition SVD d'une image:
#Ritish et Andrea
# Libraries used
library(png)
library(base)
library(BBmisc)
library(stats)
library(SpatialPack)
#%%%%%%%%%%%%%%%%%%

# Exemple de chargement d'un fichier PNG
img_mont<-readPNG("mont.png", info = TRUE)
img_mont

#On affiche l'image en plot
plot(img_mont, xlab="i", ylab = "j")

#L'image transformé a une matrice
img_matrix<-as.matrix(img_mont)
img_matrix

#On centre et réduit les valeurs
norm_image <-BBmisc::normalize(img_matrix,
method="standardize", range = c(0,1))
norm_image

#Plot de l'image normalisé
```

```
plot(norm_image,xlab="i", ylab = "j")

#SVD using the standardized matrix
# On decompose la matrice de l'image avec la
#SVD sous la forme S * SIGMA * T^T
svd_decomp<-svd(norm_image)
svd_decomp

#La Matrice S
svd_s<-svd_decomp$u
svd_s

#La Matrice diagonale Sigma des valeurs singulières
svd_sigma<-diag(svd_decomp$d)
svd_sigma

# La matrice T transpose
svd_t<-svd_decomp$v
svd_tp<-t(svd_t)
svd_tp

#La SVD multiplication
svd_matrix<-svd_s %*% svd_sigma %*% svd_tp
svd_matrix

#Equal test to check if svd matrix = norm_img matrix
Equal_test<-all.equal(norm_image,svd_matrix)
Equal_test

#Reconstruction de l'image avec la SVD: La fonction
```

qui extrait le niveau de details de l'image avec standardization

```
svd_image<-function(n)
{#LA decomposition SVD
image_final<-svd_decomp$u[,1:n] %*% diag(svd_decomp$d[1:n]) %*%
t(svd_decomp$v[,1:n])

#L'image avec la matrice de SVD. On fait une rotation de 90'

image_final_rot<-t(image_final)[,nrow(image_final):1]

#Image en gray pour avoir les valeurs entre 0 et 1
return(image_sortie<-image(image_final_rot,
col=grey(seq(0,1,length=180)),axes=FALSE))}

# On affiche l'evolution de la decomposition SVD avec
une grille de 3x3 avec 9 état de itérations

par(mfrow=c(3,3),mar=rep(4,4))
img_final_n <- sapply(c(2,20,30,50,90),svd_image)

#On change la grille d'affichage de graphique à défaut
dev.off()
par(mfrow=c(1,2))

#Reconstruction de l'image avec la SVD avec la standardization
svd_image(2)  # 110.3 kb
svd_image(20) # 147.3 kb
svd_image(30) # 150.7 kb
svd_image(50) # 150   kb
```

```
svd_image(90) # 151.3 kb

#Graphics
# On visualize les valeurs singulieres au carées sur la somme
pour voir la variance expliquée de chaque valeur singulière

plot(svd_decomp$d^2/sum(svd_decomp$d^2),type="o",
xlab="Sum of Square Sigma", ylab ="Contribution of the singular value",
col="dark red")
axis(2,col="dark red")
par(mfrow=c(1,2))

# On créer la séconde graphique
plot(x=c(1:100),y=lapply(c(1:100),FUN = function(n)
{sum(svd_decomp$d[1:n]^2)/sum(svd_decomp$d^2)}),
type="o",col="dark blue",
ylab ="Singular value contribution to image accuracy",
xlab="Sum of Sigma squared",bty="o")

#Fonctions MSE et PNSR
#La somme des erreurs carrées de notre decomposition SVD
MSE<-function(f,m)
{g<-svd_decomp$u[,1:m] %*% diag(svd_decomp$d[1:m]) %*%
t(svd_decomp$v[,1:m])
mse<-(1/(nrow(f)*m))*sum(sum(f[nrow(f):m,nrow(f):m]
-g[nrow(f):m,nrow(f):m])^2)
return(mse)}

#Peak Signal to noise ratio
PNSR<-function(f,m)
```

```
{g<-svd_decomp$u[,1:m] %*% diag(svd_decomp$d[1:m]) %*%
t(svd_decomp$v[,1:m])
mse<-(1/(nrow(f)*m))*sum(sum(f[nrow(f):m,nrow(f):m]
-g[nrow(f):m,nrow(f):m])^2)
pnsr<-20*(log10(max(f[1:m,1:m])/sqrt(mse)))
return(pnsr)}

#MSE ET PNSR for rank n

MSE(norm_image,2)
MSE(norm_image,20)
MSE(norm_image,30)
MSE(norm_image,50)
MSE(norm_image,90)

PNSR(norm_image,2)
PNSR(norm_image,20)
PNSR(norm_image,30)
PNSR(norm_image,50)
PNSR(norm_image,90)

#Graphics of MSE and PNSR

#rang de la matrice SVD
Rank_M<-c(2:100)

#Graphique du Mean square error
MSE_norm <-sapply(Rank_M,MSE,f=norm_image)
plot(x=Rank_M,y=MSE_norm,type="l",col="dark blue",
ylab ="Mean Square Error (MSE)",
```

```
xlab="Rank nxp of reduced SVD matrix",bty="o")

#Graphique de Peak Signal to Noise Ratio
PNSR_norm <- sapply(Rank_M,PNSR, f=norm_image)
plot(x=Rank_M, y=PNSR_norm, type="l", col="red",
ylab ="Peak Signal to Noise ratio (PSNR)",
xlab="Rank nxp of reduced SVD matrix",bty="o",ylim =c(-45,85))

#L'image de la matrice original A*A^T et A^T*A (Image)

original1 <-(img_matrix %*% t(img_matrix))
image(original1 , col=grey(seq(0,1, length=180)),axes=FALSE)

original2 <- (t(img_matrix) %*% img_matrix)
image(original , col=grey(seq(0,1, length=180)),axes=FALSE)

#IMAGE DECOMPOSITION without standardization

#SVD without standardization
svd_nodenoise <-svd(img_matrix)
svd_matrix_nodenoise<-svd_nodenoise$u %*% diag(svd_nodenoise$d) %*%
t(svd_nodenoise$v)
sigma_nostd<-diag(svd_nodenoise$d)

#SVD function with no standardization

svd_image2<-function(n)
{#LA decomposition SVD
image_final2<-svd_nodenoise$u[,1:n] %*% diag(svd_nodenoise$d[1:n])
%*% t(svd_nodenoise$v[, 1:n])
```

```
#L'image avec la matrice de SVD. On fait une rotation de 90'

image_final_rot2<-t(image_final2)[,nrow(image_final2):1]

#Image en gray pour avoir les valeurs entre 0 et 1
image_sortie2<-image(image_final_rot2,
col=grey(seq(0,1,length=180)),axes=FALSE)}

svd_image2(2)  # 116.2 kb
svd_image2(20) # 159.8 kb
svd_image2(30) # 161.3 kb
svd_image2(50) # 168.4 kb
svd_image2(90) # 170.4 kb

MSE2<-function(f,m)
{g<-svd_nodenoise$u[,1:m] %*% diag(svd_nodenoise$d[1:m]) %*%
t(svd_nodenoise$v[,1:m])
mse2<-(1/(nrow(f)*m))*sum(sum(f[nrow(f):m,nrow(f):m]-g[nrow(f):m,
nrow(f):m])^2)
return(mse2)}

PNSR2<-function(f,m)
{
g<-svd_nodenoise$u[,1:m] %*% diag(svd_nodenoise$d[1:m]) %*%
t(svd_nodenoise$v[,1:m])
mse2<-(1/(nrow(f)*m))*sum(sum(f[nrow(f):m,nrow(f):m]
-g[nrow(f):m,nrow(f):m])^2)
pnsr2<-20*(log10(max(f[1:m,1:m])/sqrt(mse2)))
return(pnsr2)}
```

```
Rank_M2<-c(2:100)
#Graphique du Mean square error
MSE_norm2 <-sapply(Rank_M2,MSE2,f=img_matrix)
plot(x=Rank_M2,y=MSE_norm2,type="l",col="dark blue",
ylab ="Mean Square Error (MSE)",
xlab="Rank nxp of reduced SVD matrix",bty="o")

#Graphique de Peak Signal to Noise Ratio
PNSR_norm2 <- sapply(Rank_M2,PNSR2,f=img_matrix)
plot(x=Rank_M2,y=PNSR_norm2,type="l",col="red",
ylab ="Peak Signal to Noise ratio (PSNR)",
xlab="Rank nxp of reduced SVD matrix",bty="o", ylim =c(-45,85))

#CONDITION NUMBER OF THE MATRIX
#Avant la normalisation
cond_num<-kappa(img_mont, exact = TRUE)
cond_num

#Apres la normalisation
cond_num_norm<-kappa(norm_image, exact = TRUE)
cond_num_norm

dev.off()
#Condition number function of matrix with rank 2:n
condition_number <- function(f,k)
{return(kappa(f[1:k,1:k],exact = TRUE))}
cond_range <- c(1:100)

#Condition number of matrix with rank 2:n with
```

```
image without normalised matrix

kappa_range <- sapply(cond_range,condition_number,f=sigma_nostd)
plot(x=cond_range,y=kappa_range,type="l",col="dark green",
ylab ="Condition number SVD without Standardization",
xlab="Condition number of SVD rank nxp",bty="o")

#Condition number of matrix with rank 2:n with image
of the SVD decomposition matrix with normalised matrix

kappa_range <- sapply(cond_range,condition_number,f=svd_sigma)
plot(x=cond_range,y=kappa_range,type="l",col="dark green",
ylab ="Condition number SVD with Standardization",
xlab="Condition number of SVD rank nxp",bty="o")

#Scenario analysis of the SVD standardized
matrix of the condition number

#Condition number of n singular values (Standardized)

kappa(svd_sigma[1:2,1:2], exact = TRUE)
kappa(svd_sigma[1:20,1:20], exact = TRUE)
kappa(svd_sigma[1:30,1:30], exact = TRUE)
kappa(svd_sigma[1:50,1:50], exact = TRUE)
kappa(svd_sigma[1:90,1:90], exact = TRUE)

#Condition number of n singular values (Not Standardized)

kappa(sigma_nostd[1:2,1:2], exact = TRUE)
kappa(sigma_nostd[1:20,1:20], exact = TRUE)
```

```
kappa(sigma_nostd[1:30,1:30], exact = TRUE)
kappa(sigma_nostd[1:50,1:50], exact = TRUE)
kappa(sigma_nostd[1:90,1:90], exact = TRUE)


#Condition number of AT*A using svd singular values
kappa(t(svd_sigma[1:2,1:2])%*%svd_sigma[1:2,1:2], exact = TRUE)
kappa(t(svd_sigma[1:20,1:20])%*%svd_sigma[1:20,1:20], exact = TRUE)
kappa(t(svd_sigma[1:30,1:30])%*%svd_sigma[1:30,1:30], exact = TRUE)
kappa(t(svd_sigma[1:50,1:50])%*%svd_sigma[1:50,1:50], exact = TRUE)
kappa(t(svd_sigma[1:90,1:90])%*%svd_sigma[1:90,1:90], exact = TRUE)

#Scenario Analysis
#Standardized Image matrix with SVD

MSE(norm_image,2)
MSE(norm_image,20)
MSE(norm_image,30)
MSE(norm_image,50)
MSE(norm_image,90)


PNSR(norm_image,2)
PNSR(norm_image,20)
PNSR(norm_image,30)
PNSR(norm_image,50)
PNSR(norm_image,90)

#Original Image matrix with SVD
```

```
MSE2(img_matrix,2)
MSE2(img_matrix,20)
MSE2(img_matrix,30)
MSE2(img_matrix,50)
MSE2(img_matrix,90)

PNSR2(img_matrix,2)
PNSR2(img_matrix,20)
PNSR2(img_matrix,30)
PNSR2(img_matrix,50)
PNSR2(img_matrix,90)

#COMPUTATION TIME OF SVD
#Not standardized SVD

system.time(svd_image2(2))
system.time(svd_image2(20))
system.time(svd_image2(30))
system.time(svd_image2(50))
system.time(svd_image2(90))

#Not standardized SVD

system.time(svd_image(2))
system.time(svd_image(20))
system.time(svd_image(30))
system.time(svd_image(50))
system.time(svd_image(90))

svd_image<-function(n)
```

```
{#LA decomposition SVD
image_final<-svd_decomp$u[,1:n] %*% diag(svd_decomp$d[1:n])
%*% t(svd_decomp$v[,1:n])

#L'image avec la matrice de SVD. On fait une rotation de 90'
image_final_rot<-t(image_final)[,nrow(image_final):1]

#Image en gray pour avoir les valeurs entre 0 et 1
return(image_sortie<-image(image_final_rot,col=grey(seq(0,1,length=180)),
axes=FALSE))}

svd_image2<-function(n)
{#LA decomposition SVD
image_final2<-svd_nodenoise$u[,1:n] %*%
diag(svd_nodenoise$d[1:n]) %*%
t(svd_nodenoise$v[, 1:n])

#L'image avec la matrice de SVD. On fait une rotation de 90'
image_final_rot2<-t(image_final2)[,nrow(image_final2):1]

#Image en gray pour avoir les valeurs entre 0 et 1
image_sortie2<-image(image_final_rot2,col=grey(seq(0,1,length=180)),
axes=FALSE)}
```