UNIVERSITAT DE BARCELONA

MSc FUNDAMENTAL PRINCIPLES OF DATA SCIENCE

PROJECT 2 NLA: SVD APPLICATIONS

Authors:

Jordi Segura

11 December 2022

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.

In this first part of the project, we want to solve the Least Sqaures problem, or LS problem from now on, using the QR factorization and the SVD one. We wrote 2 functions, one using directly the SVD function and the other one using the QR one. We must be careful with the QR factorization because it is not capable of holding rank-deficient matrices.

As we will see, for the first datafile it will not be a problem, but will be it in the second one. For these reasons, we have to code a version of the QR which takes into account the rank of the matrix.

Below, we find the results(Figure 1/Figure 2) and comparisons between the two factorization algorithms. The results are the same until the degree grows to 25, where the norm and the solution starts differing. We can also see this behaviour in a graphical way, see Figure 3, where both curves are overlaped except the ones for the degree 25.

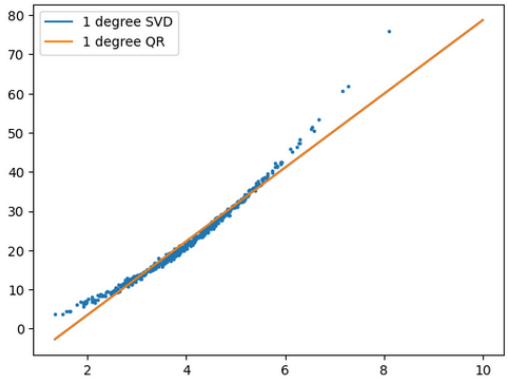
For the second datafile, we encountered the problem of the rank-deficient matrix as previously stated. Once solved, the results of both algorithms differ and, indeed, the SVD seems unstable.

```
Degree ===> 1
 SVD:
  LS Solution =  [-15.26610476   9.39864351]
  Norm =  17.92731029687858
  Time =  0.0

 QR:
  LS Solution =  [-15.26610476   9.39864351]
  Norm =  17.927310296878577
  Time =  0.0


Degree ===> 2
 SVD:
  LS Solution =  [0.20000039 1.31085632 0.98928421]
  Norm =  1.6543964762623309
  Time =  0.0

 QR:
  LS Solution =  [0.20000039 1.31085632 0.98928421]
  Norm =  1.6543964762623364
  Time =  0.0


Degree ===> 5
 SVD:
  LS Solution =  [-2.85970427e+00  5.37082597e+00 -1.00593155e+00  4.58097984e-01
 -4.94839267e-02  2.02332516e-03]
  Norm =  6.184487438522982
  Time =  0.0

 QR:
  LS Solution =  [-2.85970427e+00  5.37082597e+00 -1.00593155e+00  4.58097984e-01
 -4.94839267e-02  2.02332516e-03]
  Norm =  6.184487438514976
  Time =  0.0


Degree ===> 25
 SVD:
  LS Solution =  [ 4.17894761e-05 -2.66212234e-06  5.53743030e-08  3.00451196e-08
 -5.83557007e-09  1.75164465e-07  9.34964735e-08  3.03463658e-07
  5.58065646e-07  4.94352931e-07  5.93359731e-07  9.65305349e-07
  1.14699067e-06  1.88017005e-06  2.21199494e-06  1.31655408e-06
 -8.54459337e-07 -1.46922777e-06  1.35220976e-06 -4.91555048e-07
  9.59491951e-08 -1.02228287e-08  4.49493795e-10  1.64023399e-11
 -2.59233588e-12  8.02840385e-14]
  Norm =  4.209355581819017e-05
  Time =  0.0

 QR:
  LS Solution =  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  9.28066212e-09  0.00000000e+00
 -4.40170977e-09  2.19673935e-09 -4.92011645e-10  5.86075161e-11
 -3.62558028e-12  9.20059674e-14]
  Norm =  1.051555635649343e-08
  Time =  0.0050487518310546875
```

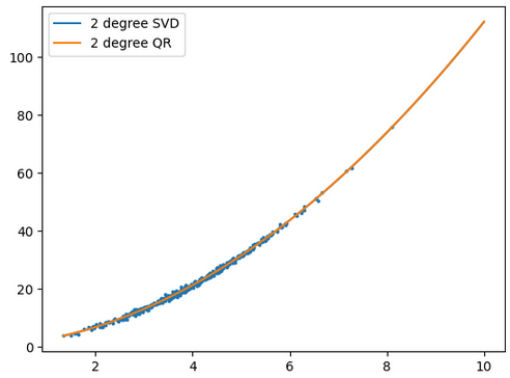
Figure 1: Results of datafile

```
************************************************
Datafile 2
************************************************
 SVD:
  LS Solution =  [ 4.63553184e+15 -4.63553184e+15 -9.52805090e+02  1.44785817e+04
 -9.71644693e+04  3.60187959e+05 -8.04451497e+05  1.11153883e+06
 -9.30293116e+05  4.31972012e+05 -8.53157274e+04]
  Norm =  6555631995485197.0
  Time =  0.0

 QR:
  LS Solution =  [ 1.66061275e+01  0.00000000e+00 -1.88268350e+03  2.99498591e+04
 -2.12104358e+05  8.37034712e+05 -2.00324191e+06  2.97903439e+06
 -2.69206452e+06  1.35366101e+06 -2.90442987e+05]
  Norm =  4774736.28643376
  Time =  0.0
```

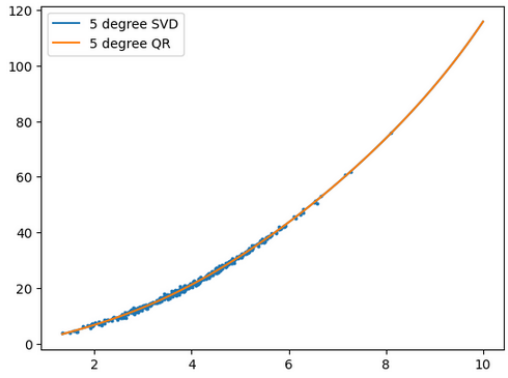
Figure 2: Results of datafile2



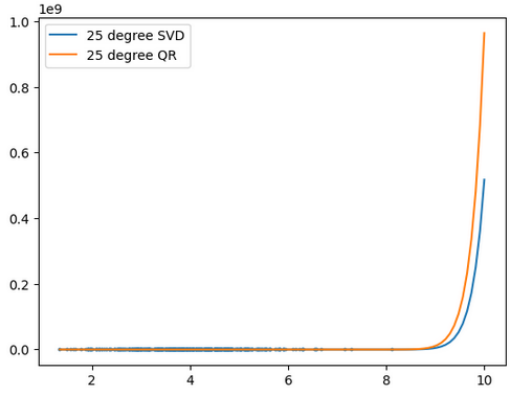
(a) Plot of the LS of a solution with degree 1



(b) Plot of the LS of a solution with degree 2



(c) Plot of the LS of a solution with degree 5



(d) Plot of the LS of a solution with degree 25

Figure 3: Plot of different times w.r.t their degrees

2 Graphics Compression

2.1 SVD - Frobenius statement

The SVD factorization has the property of giving the optimal low-rank approximation of a given matrix with respect to the Frobenius and 2-norms.

Proof for the Frobenius norm: Let A be a given matrix and let U, S, and V be its SVD factorization, where S is a diagonal matrix containing the singular values of A in non-increasing order. Let B be a low-rank approximation of A with rank r, where $0 \leq r \leq \min(m,n)$. Then the Frobenius norm of the difference between A and B can be expressed as:

$$\|A-B\|_F = \left\|U*S*V^T - B\right\|_F$$

By the triangle inequality, we have:

$$\|A-B\|_F \leq \left\|U*S*V^T - U*S_r*V_r^T\right\|_F + \left\|U*S_r*V_r^T - B\right\|_F$$

where S_r is a diagonal matrix containing the first r singular values of A, and V_r is the matrix containing the first r columns of V.

Since the SVD factorization minimizes the Frobenius norm of $U*S*V^T - B$, we have:

$$\left\|U*S*V^T - U*S_r*V_r^T\right\|_F = 0$$

Thus, we can simplify the inequality as:

$$\|A-B\|_F \leq \left\|U*S_r*V_r^T - B\right\|_F$$

Since SVD is an optimal low-rank approximation of A with respect to the Frobenius norm, we have:

$$\left\|U*S_r*V_r^T - B\right\|_F = 0$$

Therefore, we can conclude that the SVD factorization gives the optimal low-rank approximation of A with respect to the Frobenius norm.

Proof for the 2-norm: To demonstrate the optimality of the SVD factorization for the 2-norm, we can first show that the SVD factorization minimizes the 2-norm of $U*S*V^T - B$ for any low-rank approximation B with rank r. This is because the singular vectors in U and V are orthonormal, and the singular values in S are non-increasing. Therefore, the product $U*S*V^T$ is the best rank-r approximation of A in the least-squares sense, and hence minimizes the 2-norm of the difference.

Next, we can show that the SVD factorization is optimal by contradiction. Suppose there exists a low-rank approximation B' with rank r that has a smaller 2-norm than the SVD factorization $U*S_r*V_r^T$. Then we have:

$$\|A-B'\|_2 < \left\|A - U*S_r*V_r^T\right\|_2$$

By the triangle inequality, we have:

$$\|A-B'\|_2 \leq \left\|A-U*S*V^T\right\|_2 + \left\|U*S*V^T - U*S_r*V_r^T\right\|_2 + \left\|U*S_r*V_r^T - B'\right\|_2$$

Since the SVD factorization minimizes the 2-norm of $U*S*V^T - B$, we have:

$$\left\|U*S*V^T - U*S_r*V_r^T\right\|_2 = 0$$

Thus, we can simplify the inequality as:

$$\|A - B'\|_2 \leq \left\|A - U * S * V^T\right\|_2 + \left\|U * S_r * V_r^T - B'\right\|_2$$

Since B' has a smaller 2-norm than $U * S_r * V_r^T$, we have:

$$\|A - B'\|_2 < \left\|A - U * S * V^T\right\|_2 + \left\|U * S_r * V_r^T - B'\right\|_2$$

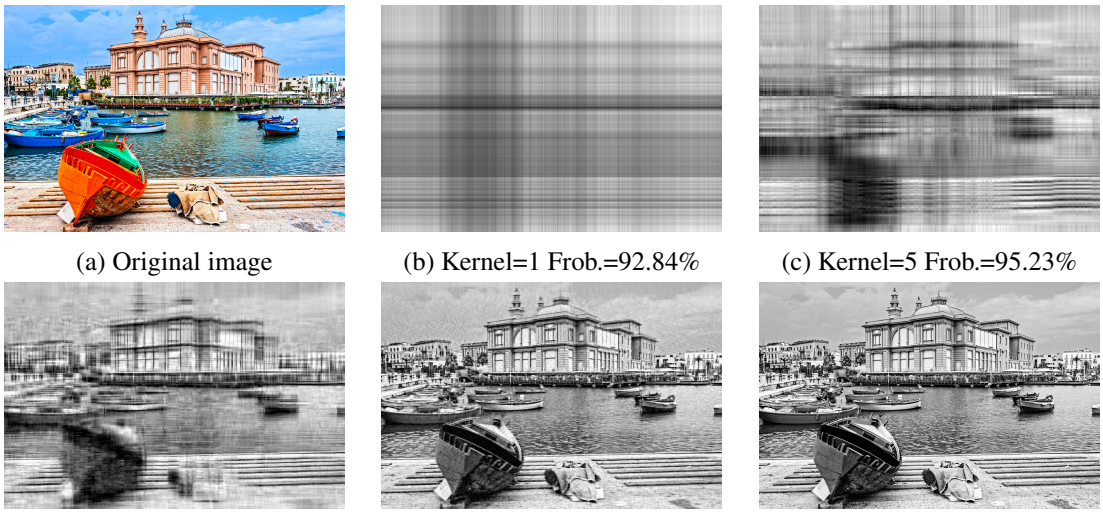
This contradicts the triangle inequality, and thus we conclude that the SVD factorization is optimal for the 2-norm.

2.2 SVD with Images

In this part of the project, we want to analyze the approximation of the lower rank images. We have used two images, one landscape in Italy and one image of the Arabic alphabet with letters, both colored images but converted to grey-scale before the compression. The function we coded returns us the Frobenius norm captured by the lossy approximation as well as the image compressed.

In the first example(Figure 4) we see how with Kernel values 1,5 the image is unrecognizable, but from 20 on, the image is clear with a Frobenius norm of >97%.

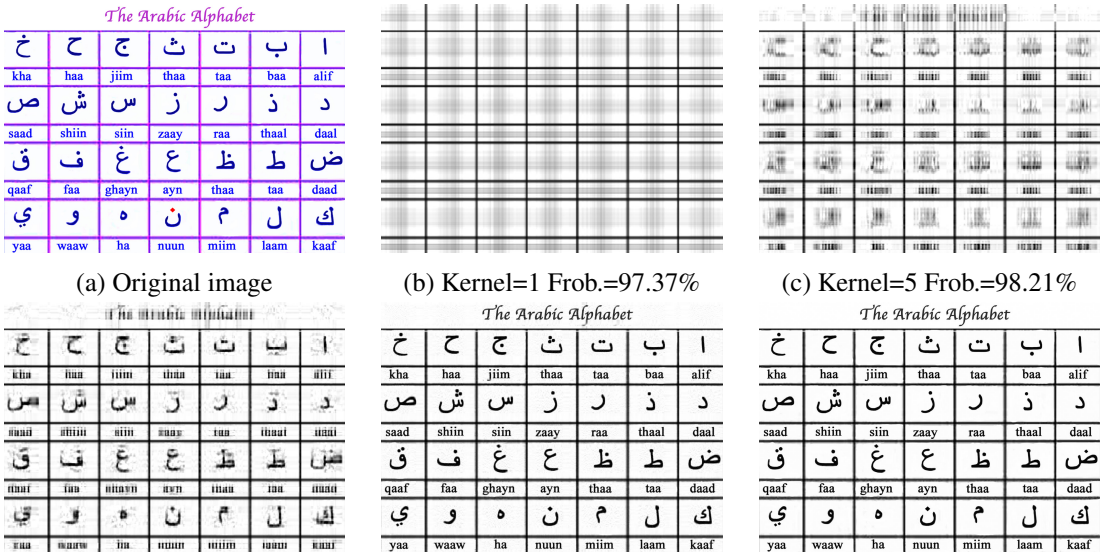
On the other hand, the second image(5 even with a Frobenius norm of 99%, some letters are unreadble. We need a kernel value up to 100 to start seeing something clearly.



(a) Original image (b) Kernel=1 Frob.=92.84% (c) Kernel=5 Frob.=95.23%

(d) Kernel=20 Frob.=97.10% (e) Kernel=100 Frob.=98.81% (f) Kernel=500 Frob.=99.82%

Figure 4: Compression of a landscape with color



(a) Original image (b) Kernel=1 Frob.=97.37% (c) Kernel=5 Frob.=98.21%

(d) Kernel=20 Frob.=99.27% (e) Kernel=100 Frob.=99.94% (f) Kernel=500 Frob.=1%

Figure 5: Compression of a picture with letters

3 Principal Component Analysis(PCA)

In this last part of the project, we want to explore the PCA on 2 different datasets. The main idea of the PCA is to detect the main components of that dataset in order to reduce it to fewer dimensions retaining the most relevant information.

The first thing we are asked to do is to compute the PCA on a example dataset, which contains 16 observations of 4 variables. We will do it using both covariance and correlation matrices, as the theory states, and later on we will compare the results between them, such as the PCA coordinates or the deviation of each of the components. The procedure to perform such a thing is pretty straightforward if we follow the definitions. First of all, for the covariance algorithm we need to have the matrix centered subtracting the mean of the observations for each variable. Once we have it, and in order to avoid numerical instabilities we can use the SVD factorization if we consider that:

$$Y = \frac{1}{\sqrt{n-1}} X^t$$

and then $Y^t Y = C_x$, which is the covariance matrix.

With Y we can compute the SVD and the properties defined hold, such as the portion of the total variance in each component is $\frac{s_i^2}{\sum_{i=1} s_i^2}$ or that the new PCA coordinates are given by $V^t X$. Therefore, we can perform PCA and extract all the information from the SVD decomposition from the covariance matrix, as well as the correlation matrix just taking into account we need to divide the matrix by its standard deviation.

If we look at the Figure 6, we can see how the results are not identical but follow a similar pattern. The first two components hold more than 80% of the variance in both cases, but the correlation method gives more importance to the PC2 in this case. Obviously, as the results are not the same for the variance in each component, the new coordinates are not same. We can also observe to the scree plot for this dataset, which shows us the variance per component(see Figure 7), and with this in our hands, we see that an elbow appears at the second component. Thus, we should only take the first two components which, again, hold more than the 80% of our variance.

On the other hand, the second dataset is a larger one, consisting on 58581 genes and 20 observations grouped by day of that observation. We will drop the column called gene because it does not give us any useful information when performing the PCA. To compute it, we must follow the same steps above when we used the covariance matrix. First, subtract the mean, compute the new Y, and perform the SVD on that one. Once we have everything, we can store the total variance for each component as well as the new coordinates for the dataset. Now that we have 20 components, it is crucial to understand which ones are important or not, based on the variance they hold. This can be easily seen using the scree plot, see Figure 8. We can identify the first 3 components as the most important ones, holding almost the 95% of the variance of the dataset, therefore the other ones could be dropped out. Furthermore, the dataset using the first 2 components is showed in the Figure 9. Using these kind of techniques one could identify hidden patterns in the data which were impossible to see in a 20-D data.

```
************************************************
Example.dat PCA
************************************************
********************
Using COVARIANCE
********************
Portion of the total variance accumulated in each of the principal components:
[0.66979347 0.15887735 0.13452316 0.03680602]

Standard deviation of each of the principal components:
[0.12495862 0.28883346 0.19855688 0.1977119  0.22544587 0.14621423
 0.33691718 0.11815706 0.18597464 0.32491633 0.2296149  0.15742355
 0.18231641 0.12177264 0.24146043 0.22075705]

Dataset in the PCA coordinates:
[[ 7.04570495 -2.18077568 -3.74428378 -0.14176797]
 [ 1.2837116  -4.8052075   2.86289547 -0.05826626]
 [-7.15846531 -1.54081679 -1.68508057 -1.88222249]
 [-6.10964391 -1.71920315 -1.74206484  2.02960634]]
********************
Using CORRELATION
********************
Portion of the total variance accumulated in each of the principal components:
[0.60757275 0.24030055 0.11616759 0.03595911]

Standard deviation of each of the principal components:
[0.12669246 0.15052832 0.17385498 0.28567774 0.27784841 0.17744438
 0.34941439 0.05449085 0.15162098 0.33318805 0.08583455 0.31194878
 0.1608974  0.15258338 0.15345285 0.2330381 ]

Dataset in the PCA coordinates:
[[ 3.260763   -0.03166671  2.31317663  0.12504286]
 [ 1.10459592 -3.83048048 -0.32683494  0.02158858]
 [-3.71979043 -0.52430335  0.80456561  1.11397456]
 [-3.63250706 -0.65632141  1.15316631 -1.0219306 ]]
```

Figure 6: Results of example.dat

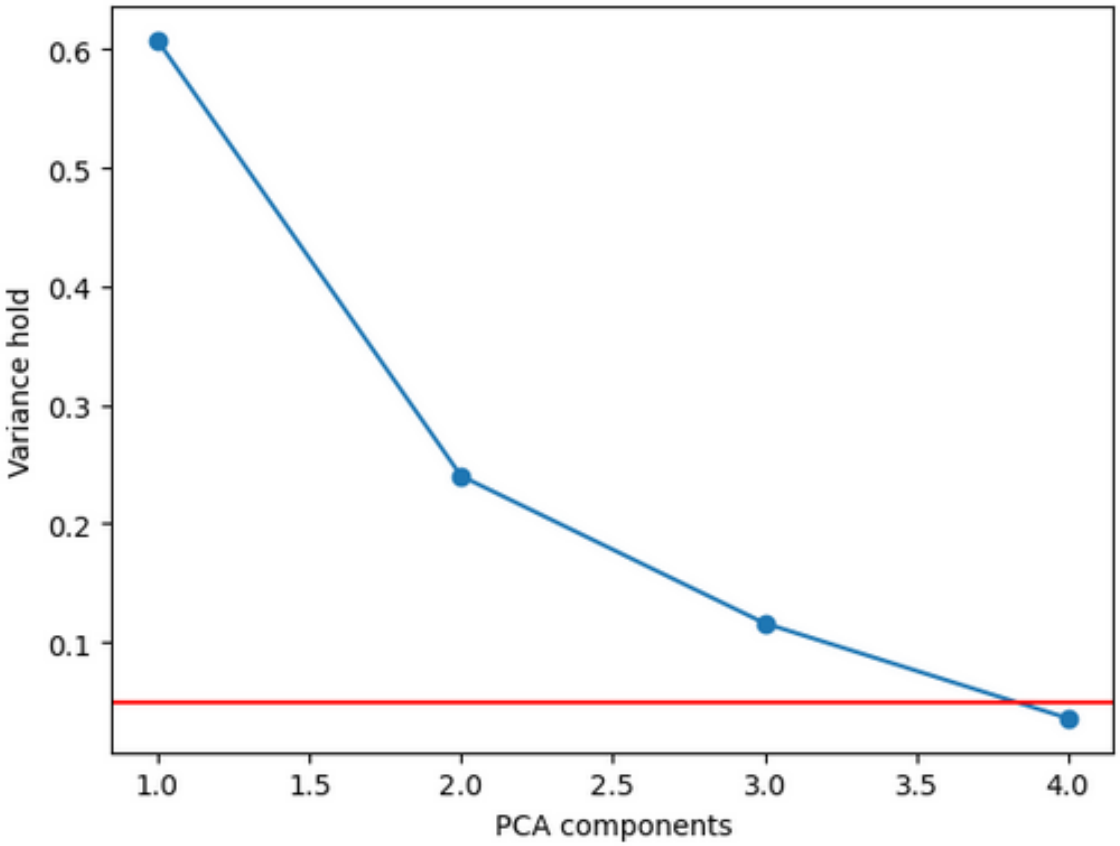


Figure 7: Scree plot of example.dat showing the variance in each component

```
**************************************************
RCsGoff.csv PCA w/ COVARIANCE
**************************************************
```

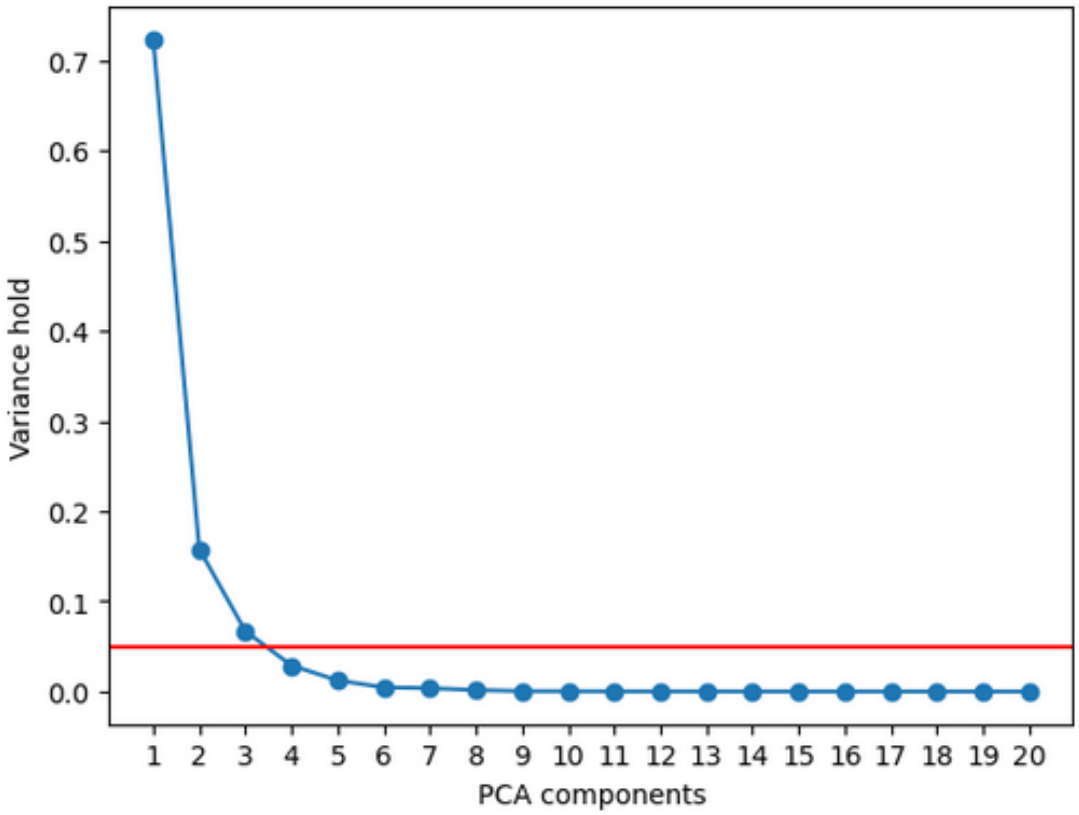


Figure 8: Scree plot of RCsGoff.csv showing the variance in each component

```
In [13]: plt.scatter(PCA_df.iloc[:,0], PCA_df.iloc[:,1])
         plt.xlabel(f'PC1 with {PCA_df.iloc[0,-1]*100} % variance')
         plt.ylabel(f'PC2 with {PCA_df.iloc[1,-1]*100} % variance')
Out[13]: Text(0, 0.5, 'PC2 with 15.784466329358269 % variance')
```

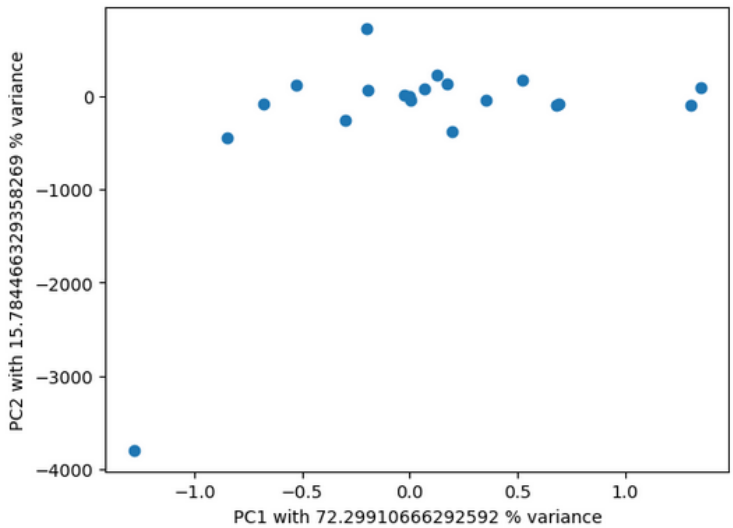


Figure 9: Plot PC1 and PC2, holding almost 90% of variance

4 Conclusions

In my opinion, this project it is quite more practical than the first one. Thanks to that, I think the coding part, once solved the mathematical procedure to implement it, it is also more straightforward.

Moreover, we can easily see the application of the PCA in our data science life, which is also important for non-mathematical people, like me, to keep in mind why we are doing these kind of projects.

Overall, I found this project useful, more practical, and has helped me to have a clearer understanding of the theory given in class about PCA, SVD and its applications.