Sorbonne Université
Faculté des Sciences et Ingénierie
Master 1 Mathématiques et applications

Année universitaire 2024–2025
MU4M053
Grands systèmes linéaires

Singular value decomposition

We cover here different uses of the SVD, either as compression tool, by taking advantage of the low-rank approximability of the data, or its "featurisability", as the dominant modes of the SVD of a matrix contain its most relevant information.

1 Image compression

In this exercise, we are going to use the SVD to compress the image `jura.jpg`. This image can be decomposed into three matrices for each primary color (red, green, blue) of size 2250×4000. The numbers contained in the matrix stand for the intensity of the associated color.

For the SVD, we are going to use the function `svd` that is in the `LinearAlgebra` package. To deal with images, we will use the `FileIO` and `Images` packages.

```julia
using FileIO, Images
img_file = load("picture.jpg") #load the picture
img_data = channelview(img_file) #transform img_file in a 3D-Array where
↪  img_data[1,:,:] contains the red part, img_data[2,:,:] contains the green part and
↪  img_data[3,:,:] contains the blue part.
img_approx = randn(size(img_data)) #or more sensible operations on img_data
img_approx_file = colorview(RGB,clamp01.(img_approx)) #transform img_approx into a
↪  writeable file, clamp01 ensures that the 3D Array has the right admissible values
save("approx.png",img_approx_file) #save the picture
```

1. Load the picture `jura.jpg` as a 3D-Array and give the SVD of the three matrices of the three colors in the image.

2. Plot the singular values in logscale in the y-axis. Comment.

3. Give the rank 10, 50, 100, 200 approximation of the original image. What is the corresponding compression and the accuracy? For the accuracy, we will use the Frobenius norm.

4. In the Frobenius norm, give the rank needed to have a 90% compression and observe the corresponding image.

Remark: in practice, the SVD is rarely used to perform image compression, as there are more powerful numerical tools for this task, relying on wavelets.

2 Feature revealing

For this exercise, we will see that the SVD can be a way to cluster data. Suppose that $A \in \mathbb{C}^{m\times n}$ is a matrix representing some data, then by the SVD, we know that if the singular values decay sufficiently fast, then for $r \ll m, n$ we have $A = \sum_{i=1}^r \sigma_i u_i v_i^*$ for some unit vectors $u_i \in \mathbb{C}^m, v_i \in \mathbb{C}^n$ and positive scalars σ_i. The first vectors $v_1, v_2, v_3, \dots$ can be used to detect patterns in rows of the matrix A. Let us consider this toy example

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$

An SVD of A is given by

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

In that case $v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$, which reflects the sparsity pattern of A.

To highlight the clustering properties of the SVD, we are going to use the MNIST handwritten digits data in the `MLDatasets` package.

```julia
using MLDatasets
train_x, train_y = MNIST.traindata() #train_x 3D Array of the images, train_y = labels
train_x[:,:,1] #matrix of the grayscale of the handwritten digits
```

1. Load the data and check for some examples in the database that the labels are correct. One can use the `heatmap` function in the `Plots` package.

   From now on, we will see the 28×28 matrix of the handwritten digits as a vector of size 28^2.

2. We are going to first filter the data to work with the handwritten digits labelled 0 and 1. From the first 5000 digits entries, retrieve the matrix $\widetilde{A}$ of the digits labelled by 0 or 1. This matrix should be of size $28^2 \times 60000$.

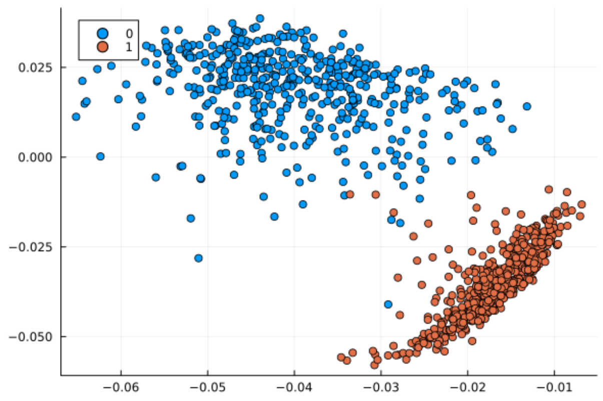
3. Recenter the data, *i.e.* the resulting $A = (a_{ij})$ should be such that $\sum_{i=1}^{28^2} a_{ij} = 0$ for all j.

4. Apply an SVD `(u,s,v)` to the matrix A.

5. Do the scatter plot `scatter(v[labels0, 1],v[labels0, 2])`, where `labels0` corresponds to the digits labelled by 0.

   Hint: look up how to manipulate logical indexing

6. Add on this plot, the scatter plot of `scatter(v[labels1, 1],v[labels1, 2])`, where `labels1` corresponds to the digits labelled by 1. You should obtain a figure that is similar to the one below.



7. Redo the exercise by adding another digit and visualise the clusters in a 2D/3D scatter plot.

3 Matrix completion

Another task where the SVD can be used is for matrix completion. The matrix completion problem is the following. Let $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ be a matrix, of which we only have access to the entries $\Omega \subset \{1,\dots,m\} \times \{1,\dots,n\}$ and where $\#\Omega \ll mn$. We want to infer the other entries of the matrix A, *i.e.* a_{ij} for $(i,j) \notin \Omega$. Without any assumption on the matrix A, it is not possible to reasonably complete this task, however, assuming that A is low-rank (and some assumption on the filling of the matrix), it is possible to retrieve a good approximation of A, in the regime where $\#\Omega \ll mn$.

For example, consider the following matrix A

$$A = \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0.3 & \cdot \\ \cdot & \cdot & 2.2 & \cdot \end{bmatrix},$$

where $\cdot$ are missing entries. If you assume that A is of rank 1, then the resulting matrix is

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.3 & 0.3 & 0.3 & 0.3 \\ 2.2 & 2.2 & 2.2 & 2.2 \end{bmatrix}.$$

The matrix completion problem is also called the *Netflix problem* where in that case, the matrix contains the ratings of movies/series by users. This matrix has a lot of missing entries as most users watch a fraction of the available catalog. The matrix completion is used to guess what are the potential movies/series that may be liked by the user.

Mathematical formulation

One way to formulate the matrix completion problem is as follows. For a given matrix $A \in \mathbb{C}^{m\times n}$, indices $\Omega \subset \{1,\dots,m\} \times \{1,\dots,n\}$ and a target rank r, we want to solve the minimisation problem

$$\min_{\substack{Z,A_r \in \mathbb{C}^{m\times n} \\ \mathrm{Rank}(A_r)\leq r \\ \forall\,(i,j)\in\Omega, Z_{ij}=A_{ij}}} \|Z - A_r\|_F.$$

It is thus a joint minimisation problem over Z and A_r where Z is the completed matrix and A_r its low-rank approximation. For a fixed Z, the minimisation problem has a solution A_r given by the truncated SVD of Z to the rank r. Now if we want to minimise $\|Z - A_r\|_F$ over Z with A_r obtained in the previous step, then we would like to set $Z = A_r$. The issue is that the matrix Z obtained with $Z = A_r$ may violate the constraint $Z_{ij} = A_{ij}$ for all $(i,j) \in \Omega$. Thus, we set the coefficients $Z_{ij} = A_{ij}$ for all $(i,j) \in \Omega$ and we go back to the beginning. This gives the following algorithm.

Input: A matrix to complete, Ω list of known indices in A, r target rank, Z initial guess, N number of iterations

```
function MATRIX_COMPLETION($A, \Omega, r, Z, N$)
    for $(i,j) \in \Omega$ do
        $Z_{ij} = A_{ij}$
    end for
    for $i = 1,\dots,N$ do
        $A_r$ = truncated SVD of $Z$ to rank $r$
        $Z = A_r$
        for $(i,j) \in \Omega$ do
            $Z_{ij} = A_{ij}$
        end for
    end for
    return $A_r$
end function
```

1. Implement the algorithm above.
2. Test it for a rank $A \in \mathbb{C}^{300\times 200}$ where A is of rank 10, and Ω a sample of 10000 indices (*i.e.* $\simeq 17\%$ of the entries) (use `sample` in the package `StatsBase` and `CartesianIndices`). Check that we retrieve a good approximation of the original matrix.

Real-life example

We are going to try the algorithm on the MovieLens 100k dataset, available in the file `u.data`. In this dataset, users have rated at least 20 movies in the database. This file contains four columns of integers:

- the first column is the user label
- the second column is the movie label
- the third column is the rating of the movie by the user
- the fourth column is irrelevant.

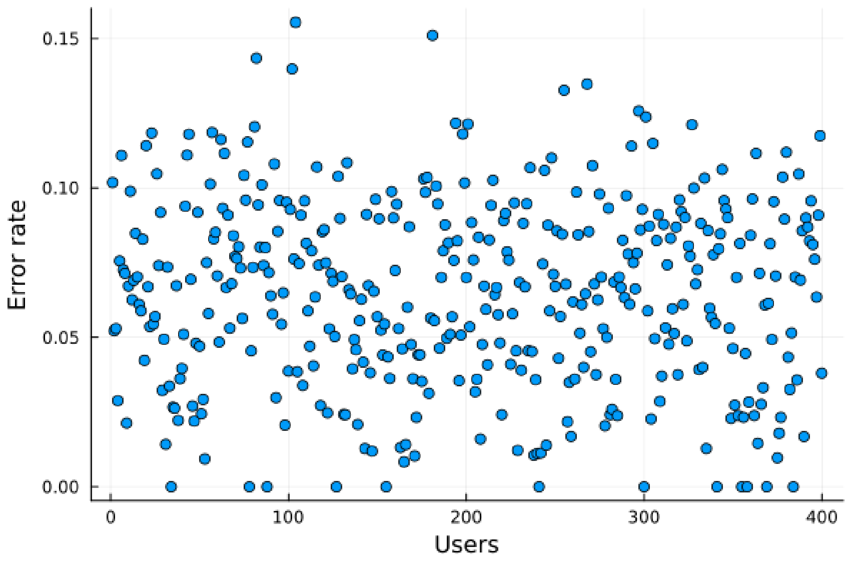
1. Write a parse function that reads the data in `u.data` and outputs a `Matrix` of size 943×1682

   Hint: one can use the `sparse` function in `SparseArrays` to build the matrix. It takes three `Arrays rows, columns, values` of the same size k such that `A=sparse(rows, columns, values)` outputs a sparse representation of the matrix A with entries `A[rows[i],columns[i]] == values[i]` for all $1 \leq i \leq k$. The other entries are null. See the example below.

```
julia> using SparseArrays
julia> rows = [1,1,2,2,3]
julia> columns = [1,3,2,3,1]
julia> values = [1.0, 2.0, -1.0, 0.5, 1.5]
julia> A = sparse(row,column,values)
3x3 SparseMatrixCSC{Float64, Int64} with 5 stored entries:
 1.0    .    2.0
  .   -1.0   0.5
 1.5    .     .
julia> Matrix(A) #convert in Matrix
3x3 Matrix{Float64}:
 1.0   0.0  2.0
 0.0  -1.0  0.5
 1.5   0.0  0.0
```

2. Test the matrix completion algorithm for the matrix restricted to the first 400 rows and with $r = 15$ (*Warning:* it may take a minute or so).

3. Let M be the initial MovieLens data matrix and M_r its low-rank approximation by completion. Compute the error rate $\frac{\|M[i,J]-M_r[i,J]\|_1}{\|M[i,J]\|_1}$ for all $1 \leq i \leq 400$, where J is the set of indices j where $M[i,j]$ has a rating. You should obtain a figure close to this one


0.15
0.10
0.05
0.00
Error rate
0
100
200
300
400
Users