Refresher course: Numerical Matrix Analysis

F. Iutzeler & J. Malick

Tutorial

A. Decompositions

○ **A.1** (Rank 1 matrices)**.**

a. Justify why a rank 1 matrix A can always be written $A = uv^{\mathrm{T}}$.
b. Express matrix $B = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 4 & 10 \\ 0 & 0 & 0 \end{bmatrix}$ as the product of two vectors: $B = uv^{\mathrm{T}}$.
c. Compute eigenvalues and associated eigenvectors of matrix B. Justify that the rank is indeed 1 using eigenvalues.
d. Show that $\det(I + uv^{\mathrm{T}}) = 1 + v^{\mathrm{T}}u$.
(hint: verify that $\begin{bmatrix} I & 0 \\ v^T & 1 \end{bmatrix} \begin{bmatrix} I + uv^T & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ -v^T & 1 \end{bmatrix} = \begin{bmatrix} I & u \\ 0 & 1 + v^T u \end{bmatrix}$*)*
e. Show that $(I + uv^{\mathrm{T}})^{-1} = I - \frac{uv^{\mathrm{T}}}{1+v^{\mathrm{T}}u}$. (This identity is called Sherman–Morrison formula)

○ **A.2** (Projection)**.** We call projection a real square matrix P such that $P = P^2$.

a. Show that if $\|P\| < 1$ for some operator norm, then $P = 0$.
b. Let P be a rank 1 matrix. Show that Px is the projection of x onto some vector. Justify why if P is symmetric, the projection is said orthogonal.
c. When $P \neq 0$ is symmetric, justify why $\|P\| = 1$.

○ **A.3** (Singular Value Decomposition)**.** Let $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

a. Compute the eigenvalues of A. Is the matrix invertible? If yes, give its inverse.
b. What are the eigenvectors corresponding to eigenvalue 2? Does A admits an eigenvalue decomposition?
c. Let $B = AA^{\mathrm{T}}$. Check that B is symmetric. What can you say about its eigenvalues? Compute the eigenvalues of B.
d. Without further computation, give the eigenvalues of $C = A^{\mathrm{T}}A$.
e. Give conditions for C to be invertible. Notably, show that if A is full column rank, then C is also full column rank.
f. Show that C^{-1} is symmetric whenever it exists. What can you then say about its eigenvalues?

B. Linear Systems Resolution with applications to Regression

In this example, that we will also study in the Labs, we use linear algebra to extract information from data; more precisely, we predict final notes of a group of student from their profiles[1].

Profiles include features such as student grades, demographic, social and school related features and were collected by using school reports and questionnaires. We wish to predict the final grade by a *good* linear combination of the features.

Mathematically, from the *learning matrix* A of size $n \times d$, $n \geq d$, comprising of the features values of each training student in line, and the vector of the values of the target features b; we seek a *regression vector* that minimizes the squared error between Ax and b. This problem boils down to the following least square problem:

$$\min_x \|Ax - b\|_2^2. \tag{B.1}$$

○ **B.1.** Using basic analysis[2], one can prove that any solution of Pb (B.1) verifies the following relation:

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b.$$

a. Assume that A has full rank, show that there is a unique solution to Pb (B.1).
b. Express this solution using the singular value decomposition of A.

In the Lab, we are going to learn a linear predictor using linear regression over a part of the data called the *learning set* and we will check our prediction by comparing the results for the rest of the data, the *testing set.*

C. PageRank

The problem of ranking webpages is of the utmost importance for search engines. To this end, a very popular approach is to represent webpages as a graph where the nodes are the pages themselves and the edges are the links between them (if page i contains a links pointing toward page j, there is a directed edge from node i to node j in the graph). Then, a page/node has a high score (it is ranked high) if there are many links pointing toward it, especially coming from highly ranked pages. This approach is at the core of the PageRank algorithm developed in 1996 by Larry Page and Sergey Brin, the founders of Google. This exercise illustrates the mathematics used in this process by working on a simple example.

More formally, consider the graph of Fig. C.1. We could choose as a score the number of incoming links; node 1 is ranked first with 3, nodes $2, 3, 4$ are second with 2, 5 is last with 1. The limits of this scoring is that $2, 3, 4$ have the same score but are very different in nature as 3 is pointed by the most important page. To correct this phenomenon, the following scoring was introduced: the score x_i of page i is equal to the sum over the pages j pointing toward i of the scores (x_j) divided by their number of outgoing links n_j. Mathematically, the (implicit) score of page i is

$$x_i = \sum_{j \in \mathcal{P}_i} \frac{x_j}{n_j} \tag{C.1}$$

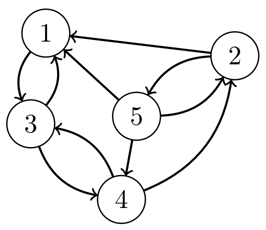
where $\mathcal{P}_i$ is the set of nodes pointing toward i (i itself is not in $\mathcal{P}_i$).

○ **C.1** (Some algebraic graph theory)**.** The graph of Fig. C.1 can be represented by its incidence matrix A which verifies $A_{i,j} = 1$ if there is a link pointing towards i in page j, and $A_{i,j} = 0$ elsewhere.

a. Let $x \in \mathbb{R}^5$ be the vector of the pages scores. Write the score equation (C.1) as a linear equation $x = Rx$ where R is a matrix to define. Does a solution to this equation exist? Is it unique?
b. Show that matrix R is column stochastic, that is, its elements are non negative and its column sum is equal to one.
c. Deduce that 1 is an eigenvalue of R. (hint: one can show that it is an eigenvalue for R^{T}.)
d. Show that $\|R\| = 1$ for some matrix norm. Deduce that the sequence $(R^n)_n$ stays bounded and that the spectral radius of R is equal to 1.

[1]The Student Performance data can be found at `https://archive.ics.uci.edu/ml/datasets/Student+Performance`; it include secondary education students of two Portuguese schools.

[2]this will be quickly covered in the optimization part.



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

FIGURE C.1. a simple graph and its incidence matrix

The above questions have led you to prove parts of a fundamental theorem in matrix analysis called the *Perron-Frobenius* theorem.

Theorem 1 (Perron-Frobenius theorem)**.** *Let A be a non-negative $n \times n$ matrix. Then,*

(i) *the spectral radius $\rho = \rho(A)$ is an eigenvalue;*
(ii) *there is a non-negative eigenvector $x \in \mathbb{R}^n_+$ such that $Ax = \rho x$;*
(iii) *if in addition A^k has all its entries (strictly) positive for some $k > 0$, then ρ is an eigenvalue of simple multiplicity and it is the only one of maximal modulus. Furthermore, there is a unique non-negative eigenvector x such that $Ax = \rho x$ and $\|x\|_1 = 1$. This vector is called the* Perron vector*.*

The additional condition of (iii) is often called *primitivity*.

Lab.

I. Introduction

To begin, we will consider square matrices (same number of lines and columns) and examine some basic operations in `Python/Numpy`.

◇ **I.1** (Basic Operations)**.** Make your first steps in matrix manipulation using `Python/Numpy` with the script `1_basic.py`.

Let $A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

a. Compute the eigenvalues of A. Compare the theoretical results with the output of `Numpy`'s eigendecomposition command `linalg.eig`.
b. Compute the eigenvalues of AA^{T}. What do you remark? What structural property enables to deduce singular values from eigenvalues?
c. Recover matrix A from the obtained decomposition. How can you compute A^{10} from this decomposition? Verify your results in the script.

◇ **I.2** (Eigendecomposition)**.** Let $A = \begin{bmatrix} 2 & -2 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix}$

a. Compute an eigendecomposition of A (*i.e.* the eigenvalues matrix, the diagonalization matrix, and check if the latter is invertible) . Without additional computations, give another diagonalization matrix.
b. Compute the eigenvalues of A^2. How are they related to the ones of A?
c. Compute the eigenvalues of $B = AA^{\mathrm{T}}$. What property ensures that B admits an eigendecomposition?
d. Compute an orthogonal diagonalization matrix of B. How can you invert this matrix in a simple manner? Verify your computation by finding B from its eigendecomposition.
e. For each matrix A, A^2, and B, check that the trace and determinant of the matrices and their eigenvalues matrix are identical. We say that the trace and the determinant are invariant by change of basis.

◇ **I.3** (Rank one matrices)**.** Examine a rank 1 matrix and its eigenvalue decomposition with the script `2_rank1.py`.

a. Compute and print the eigenvalues of the matrix given in the file. What can you notice?
b. Using ○ A.1, reconstruct A from its eigenvectors and eigenvalues. (hint: check if A is symmetric.)
c. Is the reconstruction exact?

II. Linear Systems Resolution with applications to Regression

In this example, we use linear algebra to extract information from data; more precisely, we predict final notes of a group of student from their profiles[3].

Profiles include features such as student grades, demographic, social and school related features and were collected by using school reports and questionnaires. There are $m = 395$ students (examples) and we selected $n = 27$ feature[4]. Our goal is to predict a target feature (the 28-th) which is the final grade of the student from the other features (the first 27). We assume that the final grade can be explained by a linear combination of the other features. We are going to learn from this data using linear regression

[3]The Student Performance data can be found at `https://archive.ics.uci.edu/ml/datasets/Student+Performance`; it include secondary education students of two Portuguese schools.

[4]see `student.txt` for the features description and `student-mat.csv` for the dataset.

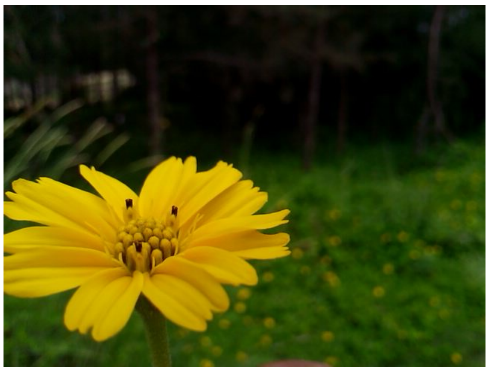


FIGURE III.1. The image manipulated in ◇ III.2.

over the $m_{learn} = 300$ students (called the *learning set*). We will check our prediction by comparing the results for the other $m_{test} = 95$ students (the *testing set*).

Mathematically, from the $m_{learn} \times (n+1)$ *learning matrix*[5] A_{learn} comprising of the features values of each training student in line, and the vector of the values of the target features b_{learn}; we seek a size-$n+1$ *regression vector* that minimizes the squared error between $A_{learn}x$ and b_{learn}. This problem boils down to the following least square problem:

$$\min_{x \in \mathbb{R}^{n+1}} \|A_{learn}x - b_{learn}\|_2^2. \tag{II.1}$$

◇ **II.1.** In `regression.py`, we take a first look at this prediction problem through Problem (II.1).

a. Observe the rank of the $m_{learn} \times (n+1)$ matrix A_{learn}. Does it have full row rank? full column rank? Conclude about the existence and uniqueness of solutions of the problem.
b. The solution of Problem (II.1) given by `Numpy`'s least square solver is given in `x_reg`. Construct the solution of the problem by using (i) the pseudo-inverse, (ii) the SVD. Check that all solutions are the same.
c. In order to test the goodness of our predictor `x_reg`, we use the rest of the data to compare our predictions with the actual observations. The test matrix A_{test} has $m_{test} = 95$ rows (students) and $n+1 = 28$ columns (features+intercept). Construct the predicted grades from `x_reg` and compare with the actual observed grades in b_{test} (`SHOW_PREDICTION = True`).
d. Compare the relative values of the coefficients of the predictor `x_reg` (`SHOW_PREDICTOR = True`). What can you observe about the relative importance of the features?

III. Singular Value Decomposition and Image Compression

The goal of this exercise is to investigate the computational aspects of the SVD; and, more importantly, observing the fact that the greater the magnitude of the singular value, the greater the importance of the associated vectors in the matrix coefficients. Investigating on this latter property will be done through the SVD of the image of Fig. III.1 seen as an array of grayscale values.

Indeed, in matrix decomposition every part is bearer of information and even though eigenvalues provide useful informations on some properties of the matrix, the associated eigenvectors are needed for a full reconstruction.

◇ **III.1.** In the script `1_svd.py`, compute the singular values of matrix A by computing (i) the eigenvalues of A^TA; (ii) the eigenvalues of AA^T. Which one is faster? Compare with (iii) `Numpy`'s `linalg.svd` command. Change the number of lines m and columns n of A. When is (i) faster than (ii)?

◇ **III.2.** In `2_image.py`, we investigate how singular values contain the information of an image.

a. Compute the singular value decomposition $U\Sigma V^{\mathrm{T}}$ of the grayscale image.

[5]the number of columns is $n+1$ as a column of ones, called *intercept*, for statistical reasons.

b. In this question, we will put `n_to_zero = 360` of the singular values[6] to zero while leaving the others unchanged; and construct new images from the modified singular values and the former matrices U and V. In `img_i`, you will put the *smallest* singular values to zero; in `img_ii`, the greatest; and in `img_ii`, random ones.
c. Observe the difference between these three modifications. What do you notice?
d. Compute the sum of the singular values for the original image and the three modified images. What can you conclude about the visual information provided by the singular values?

IV. PageRank and the power method

◇ **IV.1** (Practical computation)**.** In this part, we will compute the PageRank ordering of the graph of Fig. C.1 using the script `pagerank.py`.

a. Explain how the incidence matrix A is normalized in the code so that the sums of its columns are equal to 1.
b. Check numerically the two points shown in ∘ C.1d.
c. Iterate the matrix R a large number of times and check if the matrix is primitive. What do you notice on the eigenvalues and eigenvectors? How is defined the rank 1 matrix that you obtain? This manner of computing eigenvectors/values is called the *power method.*
d. Recover the *Perron* eigenvector of matrix R. The entries of this vector are the PageRank scores of the nodes/pages of the graph. Give the PageRank ordering of the pages of the graph.

◇ **IV.2** (To go further)**.** In this exercise, the graph we took led to a *primitive* matrix as seen above; this is necessary for the power method to work as the eigenvalue 1 has to be the only one of modulus 1. This is actually the case when the graph is strongly connected, that is when you can go from any node to any other node by following the edges, with *enough* loops. When it is not the case, our problem becomes ill posed. To overcome this problem, the ranking matrix R introduced in ∘ C.1, is replaced by

$$M = (1-\alpha)R + \alpha J, \qquad \alpha \in]0,1[$$

where is J is the 5×5 matrix whose entries are all $1/5$.
The value of α originally used by Google is 0.15.

a. Show that M is column-stochastic provided that R is.
b. Show that the problem is now well-posed.
c. Compute the ranking for the graph of Fig. C.1 but where the link from 2 to 5 is suppressed.

[6]this corresponds to 75% of the singular values