Master CMB, 1ère année
Mathematical tools for modelling, semestre 1

TP - Linear regression and image compression

Exercise 1 (Examples of linear regression problems)**.**

We investigate for the two following examples the problem of linear regression. For both examples, we have m observations y_i at different times t_i and we want to find the best line $\bar{\alpha}_0 + t\bar{\alpha}_1$ such that

$$(\bar{\alpha}_0, \bar{\alpha}_1) = \text{Argmin} J(\alpha_0, \alpha_1), \text{ with } J(\alpha_0, \alpha_1) = \sum_{i=1}^{m} |y_i - (\alpha_0 + \alpha_1 t)|^2 = \|AX - b\|^2$$

with

$$A = \begin{pmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

Pharmacokinetics of a drug The table below gives the time evolution of the concentration $(y_i)_{i=1,\dots,m}$ in lipoamide after a drug injection of 10mg at different times $(t_i)_{i=1,\dots,m}$

t_i	0	0.5	1	1.5	2	3	5	7.5	10	15	20	24
y_i	23.8	22.1	20.5	19	17.6	15.2	11.3	7.82	5.39	2.57	1.22	0.68

This concentration is usually represented by an exponential model $c(t) = c_0 e^{-kt}$, in such a way that in the log-scaled should be well approximate by a line. Here, we look for two parameters

- c_0 the initial concentration, that is linked to what is called the specific volume of the central compartment,
- k the elimination rate of the drug.

These two parameters are deeply patient dependant.

The Gompertz model in population dynamics The table below gives the temporal evolution $(y_i)_{i=1,\dots,m}$ Of the size of a tumor in the logscale for a mouse at different times $(t_i)_{i=1,\dots,m}$

t_i	6	9	13	16	20	23	27	30	34	37	
y_i	18.76	19.84	21.44	22.19	22.78	22.92	23.43	23.85	24.04	24.38	24.84

The tumor can be seen as a population of cells, so that the time evolution of the tumor size $x(t)$ can be represented thanks to a sigmoid model. The most common model used in this context is due to Benjamin Gompertz :

$$x(t) = K \left(\frac{x_0}{K}\right)^{e^{-at}},$$

in such a way that its logarithm is given by

$$y(t) = \ln(x(t)) = \ln(K) + (\ln(x_0) - \ln(K))e^{-at}.$$

The model is driven by three parameters :

- x_0 the initial size of the tumor,
- K the maximal size reachable by the tumor,
- a the growth rate of the tumor

Denote by $y_0 = \ln(x_0)$ and $b = \ln(K)$, we try to approximate y by a line

$$y(t) = b + (y_0 - b)e^{-at} \underset{t\to 0}{\sim} b + (y_0 - b)(1 - at).$$

For small time it seems reasonable, but it is the case in reality ?

Work to do

For both examples,

a. Solve the euler equations associated to this problem (also called in that case the normal equations)

```
import numpy as np
X=np.linalg.solve(?,?)
```

b. Use the function `linregress` of python

```
from scipy.stats import linregress
(a,b,rho,p,stderr)=linregress(t,y)
```

The slope of the line is given by `a`, the origin by `b`.

c. Use the function `lstsq`

```
import numpy as np
x,res,r,s=np.linalg.lstsq(A,b)
```

The output of the function `lstsq` are :

- `x` the solutions of the problem,
- `res` the residual error $\|Ax - b\|$,
- `r` the rank of A,
- `s` the singular values of A.

d. Use the method of gradient with constant step.

(a) Rewrite the algorithm as

$$X_{k+1} = C(t)X_k + \bar{b}$$

where $C(t)$ and $\bar{b}$ are a matrix and a vector to be explicited.

(b) Identify the value of t for which the algorithm converges.

(c) Identify the best value of t for which the speed of convergence is optimal.

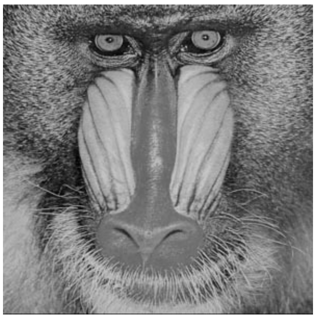
(d) Visualise the speed of convergence of the sequence.

Exercise 2 (Image compression)**.**

A possible application for the lower rank approximation of a matrix is the data compression during the transmission of a large quantity of data for example during the transmission of a very large number of images taken by a satellite. To do this, we first digitize an image by applying a n grid made of n rows and n columns. We then assign to each square a of the grid a number that corresponds to its light level of luminosity according to a scale of 0 to 256 for example. Thus the image is translated into n^2 integers that it will be necessary thereafter to transmit on earth. By proceeding in this

way, the volume of data to transmit can quickly become prohibitive when a very large number of images is involved. One possibility then consists in applying to the matrix of luminosities a singular decomposition and to transmit only the $2k$ singular vectors as well as the k singular values which that allow a satisfactory approximation of the original matrix.
Let us illustrate this compression method on two images

Fichier : lena512.png



Fichier : baboon-grayscale.jpg

For each of these image

a. Get on Ametice the file *.png or *.jpg and load on python the image (command `imread` from library `numpy`) and visualise it by using the command `imshow` of `matplotlib.pyplot` :

```
import numpy as np
import matplotlib.pyplot as plt
im1 = np.imread("lena512.png")
plt.imshow(im1,cmap=plt.cm.gray)
im2 = np.imread("baboon-grayscale.jpg")
plt.imshow(im2,cmap=plt.cm.hot)
```

Here im1 is a 512x512 matrix. Be carefull, the baboon image is a color image. More precesily, im2 is a (298,298,3) matrix. Each of the submatrix im2[:, :,i] corresponds to an intensity in RGB. You will have to convert im2 into a (298,298) matrix by taking the meanvalue of the three canal for example or taking only one of them.

b. Compute the SVD decomposition of the obtained matrices (command `svd`).

c. Give the best approximation of rank k of the matrixces and visualise the approximation.

d. For which value of k do we have a reasonable representation of the image ? What is the gain in terms of storage ?