NumPy

Fabrice Rossi

Lecture notes

NumPy arrays can be saved and loaded from files via a set of functions. Data files for this lab are available on the course web page. To read the arrays from the file `demo.npz` use

```
import numpy as np
data = np.load("demo.npz")
```

The `date` object is a dictionary like one that can be used to recover the arrays stored in `demo.npz`. Its attribute `data.files` is the list of names of those arrays. They are keys for the dictionary. For instance if the file contains 2 arrays, `A` and `B`, then `data.files` contains the list `['A', 'B']`. The arrays can be loaded by

```
array_A = data['A']
array_B = data['B']
```

When the arrays have been loaded, one should close the file via `data.close()`.

Exercise 1 (*Linear regression*)

We use in this exercise the array file `demo-linear.npz`. It contains two arrays `X` and `Y` which can be loaded as follows:

```
import numpy as np
data = np.load("demo-linear.npz")
X = data['X']
Y = data['Y']
data.close()
```

The goal of the exercise is to implement a linear regression model. We assume given a set of N observations $(x_i,y_i)_{1\leq i\leq N}$ with $x_i \in \mathbb{R}^P$ and $y_i \in \mathbb{R}$. Those observations are stored in the arrays `X` and `Y`. `X` is a matrix such that `X[i-1,j-1]` is x_{ij} and `Y` a vector such that `Y[i-1]` is y_i

We are looking for a "linear" relationship between x_i and y_i under the assumptions that

$$y_i = \sum_{j=1}^{P} \beta_j x_{ij} + \beta_0 + \epsilon_i,$$

where ϵ_i is a noise term. Given $(x_i,y_i)_{1\leq i\leq N}$ one tries to estimate $\beta = (\beta_0,\beta_1,\ldots,\beta_P)^T$. This can be done by using a least squares approach, i.e. by solving

$$\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2.$$

It can be show that the solution of this problem must fulfill the following equality

$$X^T X\beta = X^T Y, \tag{1}$$

with

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1P} \\ 1 & x_{21} & x_{22} & \dots & x_{2P} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} & \dots & x_{NP} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Question 1 Load the data and modify `X` such that it contains the matrix X (up to numbering).

Solution

```
import time
import numpy as np
data = np.load("demo-linear.npz")
X = data['X']
Y = data['Y']
data.close()
X = np.concatenate([np.full((X.shape[0],1),1),X],1)
```

A possible way to compute β is to assume $X^T X$ to be invertible and to solve equation (1) by

$$\beta = (X^T X)^{-1} X^T Y. \tag{2}$$

Question 2 Compute β using equation (2).

Solution

```
t_before = time.time()
XtX = X.T@X
# inverse matrix computation
binv = np.linalg.inv(XtX)@(X.T@Y)
t_after = time.time()
print('Elasped time:',t_after - t_before)
```

Another solution, which provides in general better results, consists in using the QR decomposition. Indeed any matrix A can be decomposed into two matrices Q and R, with $A = QR$ such that Q is orthogonal ($Q^T Q = I$) and R is upper triangular.

Question 3 Assuming $X = QR$ is the QR decomposition of X, rewrite equation (1) without using X.

Solution

We have

$$(QR)^T QR\beta = (QR)^T Y$$

and thus

$$R^T R\beta = R^T Q^T Y$$

Question 4 Assuming R is invertible, give β as the result of a simple matrix calculation.

Solution

$$\beta = (R^T)^{-1} Q^T Y$$

Question 5 Compute β using this new equation.

Solution

```
t_before = time.time()
Q, R = np.linalg.qr(X)
bqr = np.linalg.solve(R,Q.T@Y)
t_after = time.time()
print('Elasped time:',t_after - t_before)
```

Question 6 Compare the values obtained by both solutions.

Lecture notes

One can use the `time` function of the `time` module to obtain the number of seconds elapsed from a fixed reference date (January the 1st 1970 for unix computers, for instance). This can be used for crude measurements of running time as follows

```
import time
t_before = time.time()
do_something_slow()
t_after = time.time()
print('elapsed time:', t_after - t_before)
```

Question 7 Compare the running time of the solutions.

Question 8 Compute the mean squared error of the model (with one of the β estimate), that is the value of

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2.$$

Solution

```
Y_pred = X@bqr
print('MSE:',((Y_pred-Y)**2).mean())
```

Question 9 Compute the R^2 of the mode, that is the value

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2}{\sum_{i=1}^{N}\left(y_i - \frac{1}{N}\sum_{k=1}^{N} y_k\right)^2}$$

Solution

```
print('R^2:',1-((Y_pred-Y)**2).mean()/Y.var())
```