Bayesian statistics

Data reading and regression, lecture 4/1

Nicky van Foreest

April 24, 2021

Contents

1 General points

- I'll focus on coding and code/implementation
- In the lectures I intend to spend some time on general feedback on your reports and good style.
- Goal of this lecture: give you some good advice on coding. As always with good advice, this is a bit messy.

2 Info on Programming

2.1 Why do you have to program things from scratch?

What makes you different from other people?

- Any marketeer/AI/economic student can call the standard tools of python and/or R to do machine learning/data analysis/neural networks.
- When it comes to programming, there are plenty of better CS/physics/astronomy students.
- When it comes to designing and improving optimization algorithms there are plenty of better physics/math students.

You don't have an edge for standard situations, nor in the highly specific cases.

2.2 So why do you have to program?

What makes you different from other people:

- wrt marketeer/AI/economics student: You
  - understand the basics/strength/weaknesses of the standard tools.
  - make simple adaptations of the standard tools to non-standard situations
  - collaborate with math/physics students to make 'the next step.'
- wrt all the others: good knowledge of statistics

So, how do you learn to understand the strenghts and weaknesses of standard tools? By building them yourself!

2.3 Learning to program

- Can you learn to play the piano in 10 days?
- Can you learn a language (like Turkish) in 10 days?

Such claims are outright ridiculous.

- Magic bullets don't exist
- Don't read books that claim you can learn python/java/whatever. Books like that are a waste of time.
- Develop expertise, and take time to sort things out.

2.4 Programming environment and so on

- Learning a good editor pays off at the end. Why?
  - One tool for python, R, LaTeX, and so on.
  - Otherwise you have to use lots of different IDEs, and that becomes tedious
  - I use emacs. Others use notepad++.
  - Using keybindings prevents RSI, to some extent
- Jupyter with python or R, but I prefer org mode.
- Perhaps it's best to choose one language (C++?), and become real good at it.

2.5 What (not) to program, general things

In these notes I try to build many things from scratch. However, certain things I definitely don't build myself:

- classical algorithms like sorting: Your own invented algorithm for classical problems is most surely wrong, misses corner cases, and is *very* inefficient.
- Date and time algorithms: leap years, weird rules for different countries, hence extremely tricky.
- Database stuff: filtering, sorting, and so on.
- Parsing HTML/XML: use `beautiful soup`.
- Reading xls: use `pandas`

2.6 What (not) to program, numerical things

- Solving $Ax = b$, for a matrix A and vector b. This is a classical problem, which is used in many optimization problems many (millions/billions?) times per day.
- Solving LP problems.
- Advanced optimization algorithms
- random number generation: extremely tricky to find statistically sound and fast algorithms
- Many numerical algorithms, e.g., $\int_0^{10} e^{-x^4/5}\,\mathrm{d}x$.

So, I will freely use the above as black boxes. For the rest I'll try to build things from scratch.

3 Data analysis and testing

3.1 Motivation

- In the slide below I want to illustrate:
  - What to do when dealing with data
  - What to do with code.
    - Regulars want to **see** code documentation
    - They want to **see** test suites.
    - They want to **see** the output of runs of such tests.
- It is best to automatize all such boring things, and use libraries to organize it.
- Below is small demo for things I have to deal with.

3.2 Case, exam grading

Things to do:

- Get assignment grades from nestor environment for the course
- Get exam grades from nestor environment for the exam
- Get a list of students that signed up in progress. (Check)

Problems:

- Format of student numbers differs between nestor and progress.
- Not all exam participants did (all) assignments.
- Not all students appear at the exam.
- Students can have complicated names, with accents for instance.
- There are cheaters who should receive a grade.

3.3 Reading exam and assignment grades

```
def get_exam_grade():
    fname = r"gc_EXAM-EBP038A05-20210330_column_2021-04-13-15-27-45.xls"
    df = pd.read_excel(fname, sheet_name=0)  # , skiprows=0)
    number = df["Username"].apply(lambda x: int(x.replace("s", "")))
    total = df["Total"].replace(np.nan, 0)
    return {s: t for s, t in zip(number, total) if t > 0}
```

- Use the pandas library to read data
- `lambda x: int(x)` is an anonymous function. Extremely practical for one-off tasks
- Return a `dict`. Practical for look ups.
- Checks on `nan` (is 'not any number') and `t>0`.
- Reading the assignments works similar.

3.4 Computing the grade, initial code

```
def compute_grade(a, e):
    if e < 5:
        g = e
    elif a >= 6:
        g = max(0.8 * e + 0.2 * a, e)
    else:
        g = 0.8 * e + 0.2 * a

    return g
```

This appears OK, but is not. Why not?

3.5 Complications (as always)

- Do the final grades lie in $\{1, \ldots, 10\}$?
- Are grades actually computed correctly?
- Are the assignment and exam grades rounded to 1 decimal (these are intermediate grades)?
- Are the grades written to final file (for progress) equal to those that I computed?

Do tests to check! For this, use test suites.

3.6 A simple test suite

- Install `pytest`
- Make a test suite; here are two examples:

```
1 def test_min_1():
2     assert compute_grade(7, 10) > 1
3     assert compute_grade(-8, 1) == 1
4     assert compute_grade(7, 0) == 1
5
6
7 def test_max_10():
8     assert compute_grade(7, 11) == 10
```

And then a lot of such functions.

3.7 Output of the test

Read the documentation of `pytest`. Here is (some of) the output:

```
collected 2 items

compute_grades_example.py FF

FAILED ::test_min_1 - assert 0 == 1
FAILED ::test_max_10 - assert 11 == 10
```

3.8 The larger benefit of test suites

- Even if you build a small number of tests, you'll quickly spot many dumb mistakes.
- If you change (parts of) the code (refactor), you just rerun the tests, and get (some) confidence in the correctness.
- Simple cases: functions without side effects: e.g. $2 + 3$.
- Hard cases: functions with side effects (write things to file or databases) and GUIs.

3.9 Computing the grade, step 2

```
cheaters = {
    123456: "D. Dumm",
    234567: "I. Idiot ",
}

def compute_grade_final():
    final = {}
    for s in sorted(exam.keys()):
        if s in cheaters:
            continue
        e = exam[s]
        a = assignment[s]
        final[s] = compute_grade(a, e)
    return final
```

The use of dicts is really practical here. We just run over the participants of the exam, and look up the rest of the data from the different dicts.

3.10 General advice

- Use libraries like `pandas` to read data
- Use smart python data structures such as dicts, sets, lists. They help you structure your ideas, and are very efficient.
- Use test suites to check your work. In my personal work this is not extremely important. Most of my code is just to learn some concepts (research) and computations of grades and the like. Mistakes are easy to fix. My situation is not the same as yours!

4 Regression, direct approaches

4.1 Motivation/overview

- I'll illustrate different methods and numerical considerations
- I'll show strange code from the internet.
- We use this for ridge regression later

4.2 Regression without a constant

Minimize, for β:

$$f(\beta) = ||y - X\beta||^2 \tag{1}$$

Set the derivative wrt β to zero:

$$\partial_\beta f(\beta) = -2X'(y - \beta X) = 0 \tag{2}$$

$$\implies \tag{3}$$

$$\beta = (X'X)^{-1}X'y \tag{4}$$

Don't forget to repeat this yourself; it's more difficult than you might think.

4.3 Very import remarks

- Check the dimensions: $X \sim n \times p$, $y \sim n$, $\beta \sim p$, $X'X \sim p \times p$.
- Do *not* compute $(X'X)^{-1}$

Why not invert a matrix to solve $Ax = b$, even if A is invertible?

- Solving $Ax = b$ is $O(n^2)$ algorithmic complexity, solving A^{-1} is $O(n^3)$.
- The computation of A^{-1} can be numerically unstable.
- Even when A is sparse, A^{-1} can be dense. You can go from an $O(n)$ size matrix to an $O(n^2)$ size matrix. The former can fit in memory, the other not.
- See `https://www.johndcook.com/blog/2010/01/19/dont-invert-that-matrix/`

4.4 Cast in the form $Ax = b$

So, let's not solve $\beta = (X'X)^{-1}X'y$, but rewrite what we want in the form of $Ax = b$. Take

$$A = X'X, \qquad x = \beta \qquad b = X'y \tag{5}$$
$$\tag{6}$$

With this, solve for β in the system

$$Ax = b \iff X'X\beta = X'y. \tag{7}$$

4.5 Implementation, load standard libs

```
import numpy as np
from numpy.linalg import solve
import matplotlib.pyplot as plt
import pandas as pd
```

4.6 Implementation, load data and solve for β.

Let's do some analysis on song popularity on spotify. I downloaded this data set: `https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks`.

```
df = pd.read_csv("data_by_year_o.csv")
# print(df.head())  # use this to get an idea of  data

X = df[["tempo"]].values
Y = df["popularity"].values

beta = solve(X.T @ X, X.T @ Y)
print(beta)
```

```
[0.24267508]
```

Don't build this `solve` function yourself.

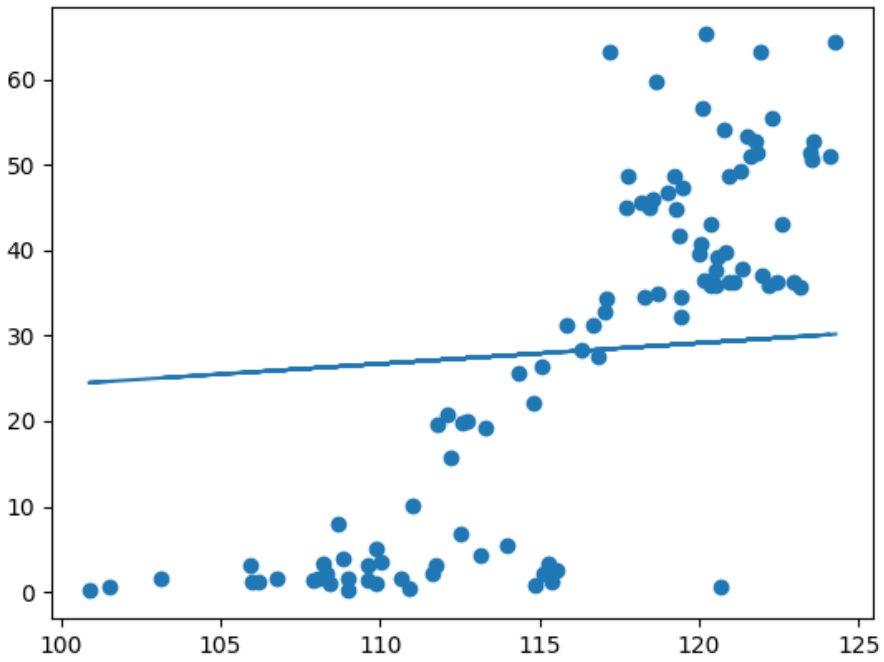
Remark. If you want to read multiple columns into X, you need a list, like this:

```
X = df[["tempo", "year"]].values

# the following gives an error
# X = df["tempo", "year"].values
```

For Y I need just one column, hence I don't need a list around "popularity".

4.7 Implementation: make plot

```
y = X @ beta

plt.clf()
plt.scatter(X, Y)
plt.plot(X, y)
# plt.show()
fname = "figures/fig_1.png"
plt.savefig(fname)
```



This is bad, why? We did not include an intercept β_0.

4.8 Regression with a constant

I found the following extension on the internet on including a constant β_0. Let's follow it. Minimize, for β_0, β:

$$f(\beta_0, \beta) = ||y - \beta_0 1 - X\beta||^2, \tag{8}$$

here $1 \sim n \times 1$.

Take derivatives wrt β_0, β:

$$\partial_{\beta_0} f = -2\, 1'(y - \beta_0 1 - X\beta) = 0, \tag{9}$$
$$\partial_{\beta} f = -2X'(y - \beta_0 1 - X\beta) = 0. \tag{10}$$

4.9 How not to turning it into $Ax = b$

We can turn the above system into the form $Ax = b$ with a bit of work.

$$1'(y - \beta_0 1 - X\beta) = 0 \implies \beta_0 + 1'X\beta = 1'y, \tag{11}$$
$$X'(y - \beta_0 1 - X\beta) = 0 \implies X'1\beta_0 + X'X\beta = X'y. \tag{12}$$

Clearly, the RHSs can be seen as a stack of $(1'y, X'y)$. With this take:

$$A = \begin{pmatrix} 1 & 1'X \\ X'1 & X'X \end{pmatrix}, \qquad x = \begin{pmatrix} \beta_0 \\ \beta \end{pmatrix}, \qquad b = \begin{pmatrix} 1'y \\ X'y \end{pmatrix}, \tag{13}$$

In my first derivation I was sloppy and did not explicitly include the $1'$ and 1. This caused me quite some trouble.

4.10 The right way to turn it into $Ax = b$

Here is a better way.

- Augment X with a column of 1's, i.e., $[1, X]$, where 1 is a vector of size n.
- With this new matrix, which we also call X, we get the results of the previous slide.
- We write β for (β_0, β) combined. (I don't like to introduce new notation when the concepts remain the same.)n

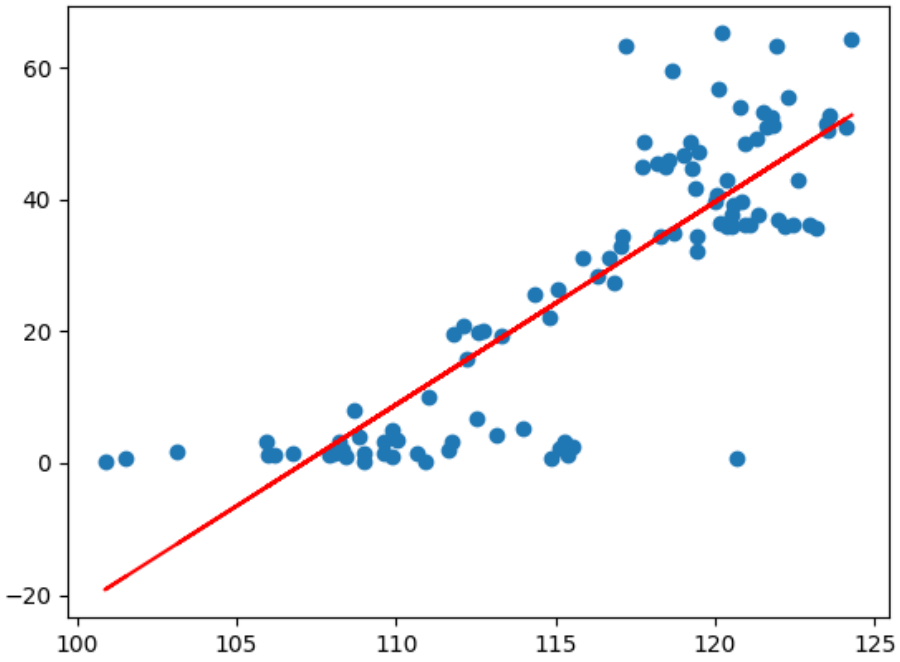
4.11 Implementation, compute β_0, β

```
xx = X  # keep for plotting
X = np.c_[np.ones(len(Y)), X] # put 1 in front

beta = solve(X.T @ X, X.T @ Y)
print(beta)
```

```
[-329.85141724    3.07913119]
```

4.12 Implementation: make plot

```
y = X @ beta

plt.clf()
plt.scatter(xx, Y)
plt.plot(xx, y, "r")
# plt.show()
fname = "figures/fig_2.png"
plt.savefig(fname)
```



Popularity of a song seems to increase with tempo.

5 General advice on coding

5.1 Good Coding:

- Try to avoid writing functions with side effects. Or, if it does, be very clear about it.
- Develop test suites. Reread above: The larger benefit of test suites', and let it sink it, and do it.

5.2 Good debugging:

- Learn to read the standard docs on the internet
- Learn to read error messages! Spend time to understand the errors you get.

6 Exercises

Read/Browse:

- pytest, and `UnitTest`, to get the logic of automatic testing, and how such tools can help you
- literature programming, to get an idea how documentation and code can be merged into one document, and that you know what 'tangling' is.
- Jupyter as a tool for literate programming. You can also use this for R, and other scripting languages. (I use orgmode, a kind of package of emacs. This surpasses jupyter in ease and power. But it's not for all people, in particular not those who want to learn power tools.)