Information on this module

Instructor Federico Poloni, `federico.poloni@unipi.it`. Office 359 DN.

Office hours (ricevimento): by appointment.

Lectures Alternating with Antonio. Irregular schedule, especially at the beginning.

Content

Antonio's part: solve all the problems: $\min_{\mathbf{x}\in S} f(\mathbf{x})$.

This part: solve a very specific problem:

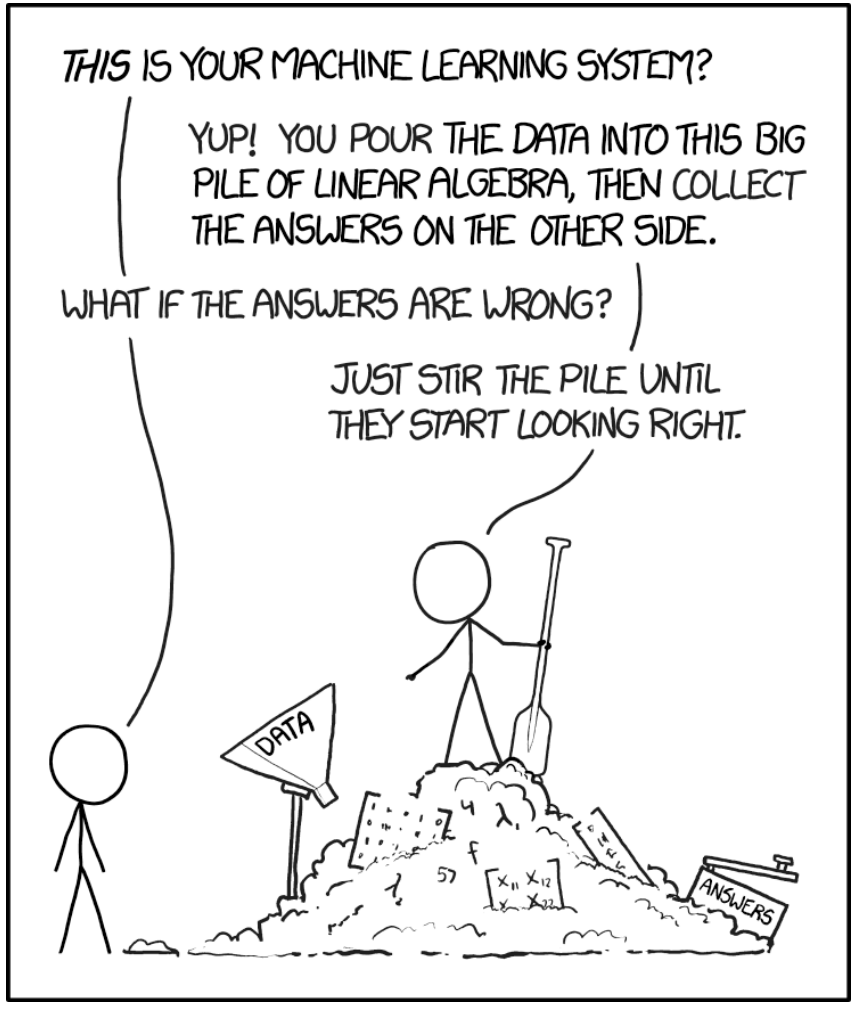
$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{y}\|_2. \tag{LS}$$

"Solve a linear system, or find the best approximation of the solution in a very specific norm"

Why is this still relevant? It is a problem we can understand well:

- Specialized algorithms that compute solutions quickly and with very high accuracy, even for very large dimensions (think $\approx 1M \times 1M$ matrices);
- We are very demanding: can we get the solution up to machine precision 10^{-16}? If no, why?
- Often appears as sub-step in more difficult problems;
- Useful for theoretical understanding: e.g., principal components.

Obligatory XKCD



https://xkcd.com/1838/

Books

Books (for this part)

1. Trefethen–Bau, *Numerical Linear Algebra*. Recommended: we will follow its approach.
2. Demmel, *Applied Numerical Linear Algebra*. Alternative source for other explanations, exercises...
3. Strang, *Linear Algebra and Learning from Data*, `https://epubs.siam.org/doi/book/10.1137/1.9780692196380`. Another alternate source to review things from a different approach. Not very detailed with proofs.
4. Eldén, *Matrix Methods in Data Mining and Pattern Recognition*, `https://epubs.siam.org/doi/book/10.1137/1.9780898718867`. Not very detailed for our purposes, but gives good insight on some topics.

2. and 3. are accessible for free from within our university network (or via VPN `https://start.unipi.it/en/help-ict/vpn/`).

Languages

Matlab: proprietary language specialized in matrix computations. Clunky language with a big Visual Basic-like IDE, great libraries and syntactic sugar for numerics.

Python: great simple language with decent numerics libraries stapled onto it.

C/C++, Fortran (yes, that's still a thing): sometimes needed for best performance.

Julia: newer, same age as Go / Rust. Stable, but tooling still not perfect. It tries to combine all advantages: syntactic sugar for numerics + libraries + fast loops.

Under the core: the same libraries (Blas/Lapack) for basic operations: summing vectors, multiplying matrices...

How to install Matlab

Available without charge to Unipi students:

- Go to `https://unipi.it/matlab`;
- Log on with your `@studenti.unipi.it` account;
- Create a Mathworks account;
- Download and install.

Suggested to save disk space and bandwidth: install only the toolboxes that you need. For this course: Matlab, Symbolic Toolbox, Optimization Toolbox.
You may be interested also in: Statistics and Machine Learning Toolbox, Neural Network Toolbox.

Alternative: Cloud Matlab `https://matlab.mathworks.com/` (with the same account).

Matlab introduction

Command window

```
>> a = 19
a =
    19
>> b = 5
b =
     5
>> a+b
ans =
    24
>> c = 2*a + b; % semicolons suppress output
```

Accuracy of operations

Default data type: `double` (IEEE standard `binary64`).

Warning: operations are only accurate 'up to 16 digits'.

```
>> (1/98) * 98 - 1
ans =
  -1.1102e-16
```

(exponential notation: stands for -1.1102×10^{-16})

Understanding the effect of these approximations will be an important part of this module.

Numerical linear algebra people are very demanding: we have "simple" problems, and we wish to solve them up to 10^{-16}, or something around there. Not 10^{-8} or 10^{-12}.