Numerical Linear Algebra

Master in Fundamental Principles
of Data Science

Teaching staff

- Martin Sombra (sombra@ub.edu)
- Arturo Vieiro (vieiro@ub.edu)

Timetable

- Theory: Wednesdays 14:10–15:00 and 15:10–16:00 (Martin)
- Practice: Fridays 16:10–17:00 and 17:10–18:00 (Arturo)

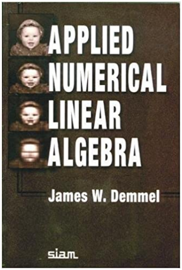
Evaluation (continuous)

- Final exam (theory): mid January 2023
- Projects (practice): during the course
- Reevaluation (if necessary): late January 2023

Prerrequisites: basic Linear Algebra

Material

- In-person classes
- Lecture notes and lists of exercices
- Books and papers



Overview

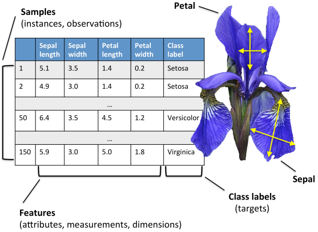
- Matrices are used to **represent data**
- Linear Algebra provides tools to understand and manipulate matrices to derive useful knowledge from data (e.g. **linear relations**)

It is a **building block** of

- Dimensionality reduction (PCA, SVD, etc)
- Machine learning (weights, loss functions, etc)
- Image processing
- Language recognition
- Etc

- Matrices are used to represent *samples* (or *data points*) with multiples *attributes* (or *variables*)
- Tipically rows correspond to samples and columns to attributes

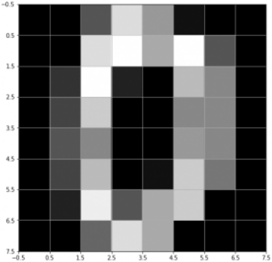
$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix}$$

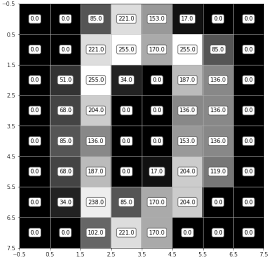


The iris flower dataset (Fischer, 1936)

Images

- A digital b/w image is made of *pixels*
- Each pixel has a value in the range 0 (black) to 255 (white)





The column span of a matrix

- Are all attributes independent?
- Can we identify the linear relationships?
- Can we reduce the size of the data matrix?

For

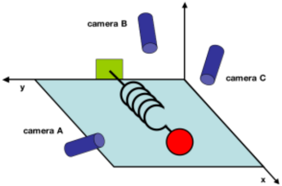
$$A = \begin{pmatrix} 1 & -1 & 0 \\ 2 & 0 & 2 \\ 3 & 2 & 5 \\ 4 & -3 & 1 \end{pmatrix}$$

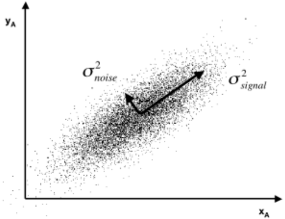
we have that $\mathrm{col}_1(A) + \mathrm{col}_2(A) - \mathrm{col}_3(A) = 0$ and so

$$\mathrm{col}_3(A) = \mathrm{col}_1(A) + \mathrm{col}_2(A)$$

The number of independent attributes equals the **rank** of the matrix

- How far is a data matrix from being *rank defective*?





Basic problems

- **Linear equation solving:** solve

$$Ax = b$$

for a nonsingular $n \times n$ matrix A, a given n-vector b, and an unknown n vector x

- **Least squares problem:** compute the n-vector x minimizing

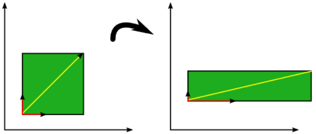
$$\|Ax - b\|_2$$

for an $m \times n$ matrix A and a given m-vector b

- **Eigenvalues and eigenvectors**, including **singular value decomposition:** find a scalar λ and a nonzero n-vector x such that

$$Ax = \lambda x$$

for an $n \times n$ matrix A



Matrix factorizations

- A **factorization** of a matrix A is its representation as a product of "simpler" matrices

Example: for an $n \times n$ matrix A, Gaussian elimination with partial pivoting (*GEPP*) computes a factorization

$$A = P\,L\,U$$

with P a *permutation*, L *unit lower triangular*, and U *upper triangular*

$$\underbrace{\begin{bmatrix} & & 1 & \\ & & & 1 \\ & 1 & & \\ 1 & & & \end{bmatrix}}_{P}\underbrace{\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & & & \\ \frac{3}{4} & 1 & & \\ \frac{1}{2} & -\frac{2}{7} & 1 & \\ \frac{1}{4} & -\frac{3}{7} & \frac{1}{3} & 1 \end{bmatrix}}_{L}\underbrace{\begin{bmatrix} 8 & 7 & 9 & 5 \\ & \frac{7}{4} & \frac{9}{4} & \frac{17}{4} \\ & & -\frac{6}{7} & -\frac{2}{7} \\ & & & \frac{2}{3} \end{bmatrix}}_{U}.$$

Solving $A\,x = b$ then breaks into three easier parts

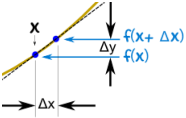
Perturbation theory and numerical stability

Two sources of **numerical errors**:

- approximations in the input data (measurements, truncations)
- errors introduced by the algorithm

  Condition numbers measure the propagation of errors

Example: let f be a real valued differentiable function. Then $f(x + \Delta x) \simeq f(x) + f'(x)\,\Delta x$, and $f'(x)$ is a condition number



Both sources of numerical errors can be unified if the algorithm is *backward stable*

How long will it take a computation?

- The complexity of an algorithm is measured in floating point operations (*flops*)

Example: GEPP solves an $n \times n$ linear system $A\,x = b$ in

$$\simeq \frac{2}{3}\,\mathrm{n}^3 \quad \text{flops}$$

Exploiting structure

- It is important to identify and exploit **special structures**, to reduce storage and increase speed

Example: when A is symmetric and positive definite, Cholesky's algorithm solves $Ax = b$ in

$$\simeq \frac{1}{3}\mathrm{n}^3 \quad \text{flops}$$

If moreover A is *banded* with band width $\sqrt{n}$, Cholesky takes only

$$O(\mathrm{n}^2) \quad \text{flops}$$

$$\mathbf{A} = \begin{pmatrix}
a_{1,1} & a_{1,2} & a_{1,3} & & & & & & & \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & & & & & & \\
a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & & & & & \\
 & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & & & & \\
 & & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & & & \\
 & & & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} & & \\
 & & & & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} & a_{7,9} & \\
 & & & & & a_{8,6} & a_{8,7} & a_{8,8} & a_{8,9} & a_{8,10} \\
 & & & & & & a_{9,7} & a_{9,8} & a_{9,9} & a_{9,10} \\
 & & & & & & & a_{10,8} & a_{10,9} & a_{10,10}
\end{pmatrix}$$

NLA in data science??

- NLA is behind most of the algorithms in data science.
- Each algorithm has its own setting, and the actual codes are versions tailored to the task at hand.
- Moreover each task involving NLA can be performed using different algorithms.

In this course we shall present some of the well-known classical NLA algorithms. We shall focus mainly on solving linear systems and computing eigenvalues of matrices (with real entries).

Furthermore, NLA is a source of interesting mathematical and computational problems.