Math 591 SS'22

First day

Jan. 18

Syllabus

- **No textbook.** My notes on Blackboard.

If you need books:

- Some concentrated on Numerical Linear Algebra:
  Ascher and Greif / Datta
- Others on NLA for data / Machine Learning:
  Aggarwal / Strang

Plenty of resources online

Large programming component.

- Python or Julia tutorials.

(No Matlab this time, unless you insist - less ML oriented)

Evaluation plan / Grading:

- Semester project. 30%

  Milestones:
  - choose by mid-February
  - Progress reports (short)
  - Presentations in last week
  - Final report (EC?)

- Midterms: 15% each
  March 10 + April 14

- Homework: 20% (biweekly)
- Quizzes: 20% (biweekly)

worst grade dropped.

I Motivation

Linear Algebra is the study of **linear operations** between **vector spaces.**

↳ Examples of spaces:

- Space of Cartesian coordinates (x, y) of all possible points in the plane w.r.t. fixed origin:

  ⇒ 2-dimensional vectors $\vec{v} \in \mathbb{R}^2$.

- Space of patients in a medical application:

  vector of attributes: age, blood sugar level, cholesterol, inflammatory markers, etc.

  Usually many attributes (the dimension.)

↳ It is common to apply linear functions to such high-dimensional vectors in application domains to extract properties.

↳ Hard to visualize geometry in 200, but same ideas!

II) Main objects:

(1) Scalars:

Individual numerical values, typically real numbers.

Example: spatial coordinate; age; temperature.

(could be rational or complex).

(2) Vectors:

Objects that can be multiplied by scalars and added together.

Typically represented by **arrays of scalars.**

Array entries may also be called coordinates, components, features (ML), etc...

Ex: vector representing data about a 25-yr old earning 30 $/hour, with 5 yrs of experience:

$$\vec{x} = [25 \quad 30 \quad 5] \in \mathbb{R}^3$$

↑ Age ↑ Salary ↑ Experience.

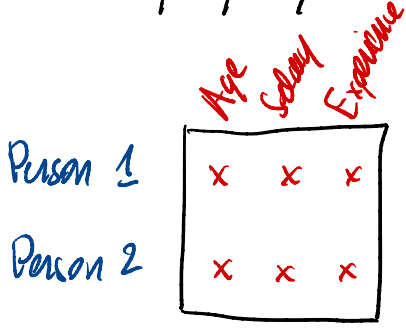
③ Matrices:

Rectangular arrays of numerical values.
Each entry is accessed by specifying its row and column index.

Example: data table containing d properties about n individuals.

⇒ each individual's properties → one row,

⇒ each property → one column.



Size $n \times d$.

⇒ Matrix $A = [a_{ij}]_{n \times d} \in \mathbb{R}^{n \times d}$.

means (i,j) entry of A denoted a_{ij}.

Note: Scalars can be viewed as vectors of size 1, or matrices of size 1×1.
Vectors of size d can be seen as matrices, either as row vectors of size $1 \times d$ or column vectors of size $d \times 1$.
(Default: column shape.)

III Basic Operations:

① Vectors:

$$\vec{x} = [x_1, \dots, x_d] \qquad a, b \text{ scalars.}$$
$$\vec{y} = [y_1, \dots, y_d]$$

Scaling: $a\vec{x} = [ax_1 \dots ax_d]$

We form **linear combinations**

$$\Rightarrow a\vec{x} + b\vec{y} = [ax_1 + by_1 \quad \dots \quad ax_d + bx_d]$$

Dot products

$$\vec{x} \cdot \vec{y} = x_1 y_1 + \dots + x_d y_d = \sum_{i=1}^{d} x_i y_i$$

Euclidean norm

$$\|\vec{x}\| = \sqrt{\vec{x} \cdot \vec{x}} = \left(\sum_{i=1}^{d} x_i^2 \right)^{1/2}$$

Note: unit vector such that $\|\vec{x}\| = 1$

Ex: $\vec{x} = \vec{v} / \|\vec{v}\| := \frac{1}{\|\vec{v}\|} \vec{v}$.

Implementation:

Python → numpy package.

```
» import numpy as np
» x = np.array([25, 30, 5])
» x.ndim
    1
» x.size
    3
» y = np.zeros(3)
» y.ndim
    1
» y.size
    3
» z = 2*x + 3*y
» np.linalg.norm(x)
    39.37...
```

Julia → native.

```
» x = [25, 30, 5]
» y = zeros(3)
» z = 2x + 3y

» using LinearAlgebra
» norm(x)
    39.37...
```

January 20th

Review of Linear Algebra.

(2) Matrices.

- **Transpose:** $A = [a_{ij}]_{m\times n}$

$$A^T = [a_{ji}]_{n\times m} \qquad A^* = [\overline{a_{ji}}]_{n\times m}$$

Example:
$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 0 & 3 \\ 1 & 4 \\ 2 & 5 \end{bmatrix}$$

Note
- $(A^T)^T = A$.
- Row vectors $\xleftrightarrow{\text{transpose}}$ Column vectors

- **Linear combination** (and addition):

if A, B have same size $m\times n$,

$$cA + dB = [ca_{ij} + db_{ij}]_{m\times n}.$$

- **Matrix-Vector product.**

An $m\times n$ matrix may be multiplied
- on the right by a (column) vector of size n,

$$Ax$$

by taking m dot products between the rows of A and x, which have size n:

$\Rightarrow$ Ax has size m.

Ex: $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \\ a_{31}x_1 + a_{32}x_2 \end{bmatrix}$

- On the left by a (row) vector y of size m, resulting in a (row) vector yA of size n.

⚠ Clearly $xA \neq Ax$ (even if both defined!)

Special case: $\vec{x} \cdot \vec{x} = (\vec{x})^T \vec{x}$

Matrix-Matrix multiplication:

$A = [a_{ij}]_{m \times k}$, $B = [b_{ij}]_{k \times n}$,

Then $C = AB$ is defined with size $m \times n$

$$c_{ij} = \sum_{r=1}^{k} a_{ir} b_{rj}$$

↳ dot product of i-th row of A, j-th column of B
(One possible interpretation!)

Properties: $A(BC) = (AB)C$, $A(B+C) = AB + AC$,
$(AB)^T = B^T A^T$, $(A+B)C = AC + BC$

⚠ $AB \neq BA$ (in general)

Outer product: form a matrix out of two (column) vectors

$u \in \mathbb{R}^m, v \in \mathbb{R}^n \quad u \otimes v = uv^T = [u_i v_j]_{m \times n} \in \mathbb{R}^{m \times n}$

Special structures:

- Zero matrix $0_{m \times n}$,
- Identity matrix $I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$

  i.e. j-th column of I_n is the basis vector $e_j = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ j^{th} row
- Symmetric matrices: $A^T = A$, Hermitian: $A^* = A$
- Triangular or trapezoidal matrices:
  - $R \in \mathbb{R}^{m \times n}$ Upper triangular: $r_{ij} = 0, \; i > j$

Ex: $m > n$, $\begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \\ 0 & 0 \end{bmatrix}$

$m = n$, $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$

$m < n$, $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \end{bmatrix}$

  - $L \in \mathbb{R}^{m \times n}$ Lower triangular: $l_{ij} = 0, \; i < j$
  - $D \in \mathbb{R}^{m \times n}$ diagonal: $d_{ij} = 0, \; i \neq j$

Essential problems of Linear Algebra

Linear System: $Ax = b$

Eigenvalue problem: $Ax = \lambda x$

Singular value problem: $Au = \sigma v$

Minimize $\|Ax\|^2 / \|x\|^2$

Factor the matrix $A = \begin{cases} LU \\ VDV^{-1} \\ U\Sigma V^* \end{cases}$

Need to understand each problem.

- $Ax = b \leftrightarrow$ Is b in the column space of A?
- $Ax = \lambda x \leftrightarrow$ Ax same direction as x?
- $Av = \sigma u$ and $A^T u = \sigma v$: Singular Values

Fundamental – allows to find the most important pieces $\sigma u v^T$ of A.

(u: column, v^T: row)

January 25 — Review of Linear Algebra

Essential problems of Linear Algebra

- Linear System: $Ax = b$
- Eigenvalue problem: $Ax = \lambda x$
- Singular value problem: $Av = \sigma u$
- Minimize $\|Ax\|^2 / \|x\|^2$
- Factor the matrix $A = \begin{cases} LU \\ V D V^{-1} \\ U \Sigma V^* \end{cases}$

Need to understand each problem.

- $Ax = b \leftrightarrow$ Is b in the column space of A?
- $Ax = \lambda x \leftrightarrow Ax$ same direction as x?
- $Av = \sigma u$ and $A^T u = \sigma v$: Singular Values

Fundamental – allows to find the most important pieces $\sigma u v^T$ of A.

u: column, v: row

It's all about spaces!

I) Linear Transformations

A matrix $A \in \mathbb{F}^{m\times n}$ can be identified with a linear map $A: \mathbb{F}^n \longrightarrow \mathbb{F}^m$, $x \longmapsto Ax$.

↝ **Important spaces:**

- The **range or column space** of A.

$$\operatorname{Ran} A = \{ Ax ;\ x \in \mathbb{F}^n \} \subset \mathbb{F}^m.$$

- The **row space** of A: $\operatorname{Ran} A^T = \{ (xA)^T,\ x \in \mathbb{F}^m \} \subset \mathbb{F}^n$.

New interpretation of the matrix–vector product:

organize A by columns $\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$ with $a_i \in \mathbb{F}^m$. Then

$$Ax = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n.$$

↳ Linear combination of columns of A!

↳ The range of A is the set of all possible linear combinations of the columns of A.

Geometry

Let's pick a 3×2 matrix $A = \begin{bmatrix} a_1 & a_2 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 4 \\ 3 & 7 \end{bmatrix}$

What part of $\mathbb{R}^3$ is formed by the column space $\{Ax = x_1 a_1 + x_2 a_2\}$?

↳ Can't be all of $\mathbb{R}^3$. It's a plane

containing
- a line in direction a_1,
- another in direction a_2.

What's more: $b = [b_1 \; b_2 \; b_3]$ is in Ran A exactly when $Ax = b$ has a solution $[x_1, x_2]$.

For example, $b = [1 \; 1 \; 1]$ is not in Ran A.

(exercise!)

In general, the possible subspaces of $\mathbb{R}^3$ could be...

- Dimension 0 : the zero vector only $\{(0,0,0)\}$
- Dimension 1 : lines of vectors $x_1 a_1$
- Dimension 2 : planes of vectors $x_1 a_1 + x_2 a_2$
- Dimension 3 : All of $\mathbb{R}^3$ $x_1 a_1 + x_2 a_2 + x_3 a_3$

We need here a_1, a_2, a_3 to be **linearly independent**

i.e. $x_1 a_1 + x_2 a_2 + x_3 a_3 = 0 \Rightarrow x_1 = x_2 = x_3 = 0$

Example: $B = [a_1 \; a_2 \; b] = \begin{bmatrix} 2 & 3 & 1 \\ 2 & 4 & 1 \\ 3 & 7 & 1 \end{bmatrix}$

$\Rightarrow$ Ran B is the whole space $\mathbb{R}^3$.

B is invertible: the problem

$$Bx = c$$

has a unique solution $x = B^{-1}c$.

Core concepts of linear algebra of spaces:

Vectors $a_1, \dots, a_n \in V$ are ...

- Linear independent if $\sum_{i=1}^{n} x_i a_i = 0 \Rightarrow x_i = 0 \; \forall i$
- Generating if any $v \in V_n$ may be written as $v = \sum_{i=1}^{n} x_i a_i$
- A basis if it is both linearly independent and generating.

Fundamental theorem: All basis of V have the same number of elements: the dimension of V.

Definition: the rank of A is the dimension of its column space:

$$\text{rank } A = \dim \text{Ran } A$$

Note that any subspace $V \subset \mathbb{R}^n$ with dimension r is also the column space of some $n \times r$ matrix

$$A = \begin{bmatrix} a_1 & \dots & a_r \end{bmatrix}$$

where $a_1, \dots, a_r$ is any basis of V.

Application: a first factorization of $A \in \mathbb{R}^{n \times n}$.

We pick a basis of $\text{Ran } A$ among its columns and arrange them into a matrix

$$C = \begin{bmatrix} a_{i_1} & \dots & a_{i_r} \end{bmatrix}$$

where $r = \text{rank } A = \text{rank } C$.

Examples:

$$A = \begin{bmatrix} 1 & 3 & 8 \\ 1 & 2 & 6 \\ 0 & 1 & 2 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 5 \\ 1 & 2 & 5 \\ 1 & 2 & 5 \end{bmatrix} \qquad C = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The "missing columns" in A can be reconstructed as linear combinations of columns of C!

Example:

coeffs of columns of A in basis of columns of C.

$$A = \begin{bmatrix} 1 & 3 & 8 \\ 1 & 2 & 6 \\ 0 & 1 & 2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix}}_{R}$$

(R is actually the row-reduced echelon form of A, without zero rows.)

Thursday Jan 27 — Linear Algebra lab

II | Four Fundamental Subspaces

Given a matrix $A \in \mathbb{R}^{m \times n}$ the following subspaces are associated with A:

① Range or Column space: $\text{Ran } A$

② Row space $\text{Ran } A^T$: linear combinations of rows

$$\{ A^T y \quad \text{for } y \in \mathbb{R}^m \}$$

③ Kernel or null-space: solutions to $Ax = 0$

$$\text{Ker } A = \{ x \in \mathbb{R}^n : Ax = 0 \}$$

④ Left nullspace: solutions to $A^T x = 0$

$$\text{Ker } A^T = \{ x \in \mathbb{R}^m : A^T x = 0 \}$$

The number of independent columns equals the number of independent rows

$$\text{rank } A = \dim \text{Ran } A = \dim \text{Ran } A^T$$

Column space — Row space

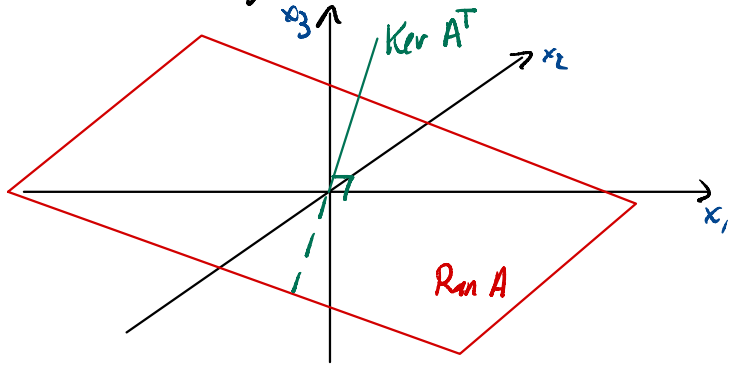
Note: (1) and (4) are subsets of $\mathbb{R}^m$
(2) and (3) are subsets of $\mathbb{R}^n$

↳ Column space is formed of linear combinations of columns,

and
the left null-space $\operatorname{Ker} A^T$ are all vectors orthogonal to the columns of A:

$$A^T x = \begin{bmatrix} a_1 \cdots a_n \end{bmatrix}^T x = \begin{bmatrix} a_1^T x & a_n^T x \end{bmatrix}.$$

These are orthogonal subspaces.



Hence

$$\operatorname{Ker} A^T = (\operatorname{Ran} A)^\perp$$

$$\dim \operatorname{Ker} A^T + \underbrace{\dim \operatorname{Ran} A}_{\text{Rank}} = m$$

Similarly the row space is orthogonal to the kernel of A:

$$\operatorname{Ker} A = (\operatorname{Ran} A^T)^\perp$$

$$\dim \operatorname{Ker} A + \underbrace{\dim \operatorname{Ran} A^T}_{\text{rank}} = n$$

Hence the dimension of all 4 fundamental subspaces can be deduced from the rank:

- $r = \dim \operatorname{Ran} A = \dim \operatorname{Ran} A^T = \operatorname{rank} A$
- $\dim \operatorname{Ker} A = n - r$
- $\dim \operatorname{Ker} A^T = m - r$

February 1 — Gaussian Elimination, LU decomposition

Fundamental problem: solve $Ax = b$.
Usually square matrix $A \in \mathbb{R}^{n \times n}$.

Unique solution exists if $\text{Rank } A = n$ (independent columns)

Typical method: Gaussian elimination.

$$\begin{cases} a_{11}x_1 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + \dots + a_{2n}x_n = b_2 \\ \dots \\ a_{n1}x_n + \dots + a_{nn}x_n = b_n \end{cases}$$

Assuming $a_{11} \neq 0$, multiply row 1 by multipliers $l_{21} = a_{21}/a_{11}$, …, $l_{n1} = a_{n1}/a_{11}$

and subtract from respective rows:

$$\begin{aligned} & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ - \quad & (a_{22} - l_{21}a_{12})x_2 + \dots + (a_{2n} - l_{21}a_{1n})x_n = b_2 \\ & \dots \\ - \quad & (a_{n2} - l_{n1}a_{12})x_2 + \dots + (a_{nn} - l_{n1}a_{1n})x_n = b_n \end{aligned}$$

Then continue to find $x_2, \dots, x_n$ using last rows, until remaining 1×1 system for x_n.

Formalization in terms of matrices.

$$A = \begin{bmatrix} 1 \times \text{row 1} \\ \ell_{21} \times \text{row 1} \\ \vdots \\ \ell_{n1} \times \text{row 1} \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \\ 0 & & \end{bmatrix} \quad \boxed{A_2}$$

rank-1 matrix $\ell_1 u_1^T$ where u_1 1st row of A,

and $\ell_1 = \begin{bmatrix} 1 \\ \ell_{21} \\ \vdots \\ \ell_{n1} \end{bmatrix}$.

Continuing the process, we have

$$A = \ell_1 u_1^T + \begin{bmatrix} 0 \times u_2 \\ 1 \times u_2 \\ \ell_{32} \times u_2 \\ \vdots \\ \ell_{n2} \times u_2 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & 0 & \\ 0 & 0 & \end{bmatrix} \quad \boxed{A_3}$$

where $u_2 =$ 2nd pivot row of $A - \ell_1 u_1^T$

and $\ell_2 = \begin{bmatrix} 0 \\ 1 \\ \ell_{32} \\ \vdots \\ \ell_{n2} \end{bmatrix}$

and after n steps,

$$A = \ell_1 u_1^T + \ell_2 u_2^T + \cdots + \ell_n u_n^T$$

$$= \begin{bmatrix} \ell_1 & \cdots & \ell_n \end{bmatrix} \begin{bmatrix} u_1^T \\ \vdots \\ u_n^T \end{bmatrix} = LU$$

lower triangular, 1's on diagonal

upper triangular

Elementary Gauss transformation matrices.

Note how 1st row of A may be obtained as $u_1^T = e_1^T A$

$$\rightarrow A^{(2)} = \begin{bmatrix} u_1 \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \Big| \; A_2 = (I - m_1 e_1^T) A.$$

where $m_1 = \begin{bmatrix} 0 \\ \ell_{21} \\ \vdots \\ \ell_{n1} \end{bmatrix}$

Definition. An elementary lower triangular matrix of order n has the form

$$M_k = I_n - m_k e_k^T = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1} & \ddots & \\ & & \vdots & & \\ & & -m_n & 0 & 1 \end{bmatrix}$$

(k^{th} column)

where $m_k = [\, 0 \; \cdots \; 0, \; m_{k+1}, \; \ldots, \; m_n \,]^T$ column vector where first k entries vanish.

Property. Let $M_k = I - m_k e_k^T$ as above then

(1) $M_k^{-1} = I_n + m_k e_k^T$

(2) $M_1^{-1} \cdots M_k^{-1} = I_n + m_1 e_1^T + \cdots + m_k e_k^T$ for $1 \le k < n$

Proof (a) Direct computation:

$$M_k (I_n + m_k e_k^T) = (I_n - m_k e_k^T)(I_n + m_k e_k^T)$$
$$= I_n - m_k e_k^T + m_k e_k^T - m_k (e_k^T m_k) e_k^T$$

Since k^{th} entry of m_k is zero, $e_k^T m_k = 0$.

This shows $M_k^{-1} = I_n + m_k e_k^T$.

(b) By induction: $M_1^{-1} \cdots M_k^{-1} = I_n + m_1 e_1^T + \cdots + m_k e_k^T$ for $1 \le k < n$. Trivial for $k = 1$. Then

$$(M_1^{-1} \cdots M_k^{-1}) M_{k+1}^{-1} = (I_n + m_1 e_1^T + \cdots + m_k e_k^T)(I_n + m_{k+1} e_{k+1}^T)$$
$$= I_n + m_1 e_1^T + \cdots + m_k e_k^T + m_{k+1} e_{k+1}^T + m_1 e_1^T m_{k+1} e_{k+1}^T + \cdots + m_k e_k^T m_{k+1} e_{k+1}^T$$

$\square$

These elementary matrices are used to create zeros in vectors! Indeed, let $a_k = [\, a_{1k} \; \cdots \; a_{nk} \,]^T$ (k^{th} column of A)

with $a_{kk} \neq 0$, $m_k = [\, \underbrace{0 \; \cdots \; 0}_{k \text{ zeros}} \; \frac{a_{k+1,k}}{a_{kk}} \; \cdots \; \frac{a_{nk}}{a_{kk}} \,]^T$

and $M_k = I_n - m_k e_k^T$.

Then $M_k a_k = [\, a_{1k} \cdots a_{kk} \; 0 \cdots 0 \,]^T$

all entries below the k^{th} have been eliminated.

Formalizing the steps of the GEM.

Starting from step k with $A^{(k)}$ of the form

$$A^{(k)} = \begin{bmatrix} u_{11} & & & & u_{1n} \\ 0 & u_{2n} & & & u_{2n} \\ \vdots & \ddots & 0 & a_{kk}^{(k)} \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} \cdots & a_{kn}^{(k)} \end{bmatrix}$$

with $A^{(1)} = A$, we form

$$M_k = I_n - m_k e_k^T \quad \text{where} \quad m_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_{k+1,k}^{(k)} / a_{kk}^{(k)} \\ \vdots \\ a_{nk}^{(k)} / a_{kk}^{(k)} \end{bmatrix}$$

and form $A^{(k+1)} = M_k A^{(k)}$ with the right form.

At the end of the process $U = M_n \cdots M_1 A$ is upper triangular and

$$A = (M_n \cdots M_1)^{-1} U = \underbrace{(I + m_1 e_1^T + \cdots + m_{n-1} e_{n-1}^T)}_{L} U$$

where $L = \begin{bmatrix} 1 & & \\ m_{21} & \ddots & \\ \vdots & \ddots & \\ m_{n1} & \cdots & m_{n,n-1} \; 1 \end{bmatrix}$ lower triangular.

⚠ Possible iff the pivot $a_{kk}^{(k)}$ is not zero!

Definition (LU factorization)

$$A = LU$$

where $L = M_1^{-1} \cdots M_{n-1}^{-1} = I_n + m_1 e_1^T + \cdots + m_{n-1} e_{n-1}^T$ is a unit lower triangular matrix with diagonal entries equal to 1, and $U = M_{n-1} \cdots M_1 A$ is upper triangular.

This requires that the pivots $u_{11} = a_{11}$, $u_{22} = a_{22}^{(2)}$, ... are all non-zero.

LU Factorization algorithm.

(1) Set $U = A$, $L = I$.
(2) For $k = 1, \ldots, n-1$ and $i = k+1, \ldots, n$
    $L[i,k] = U[i,k] / U[k,k]$
    $U[i,k] = 0$
    For $j = k+1, \ldots, n$
        $U[i,j] = U[i,j] - L[i,k] * U[k,j]$
    End
End
(3) Return L, U

Operation count $\sim 2n^3/3$

Thursday, Feb. 3 — LU Factorization and Pivoting.

I) Solution to $Ax = b$.

- Direct: $\underbrace{M_{n-1} \cdots M_1 A}_{U} x = \overbrace{M_{n-1} \cdots M_1 b}^{c}$

  ↳ Applying the same row operations on b leads to a triangular system $Ux = c$ which we solve by back-substitution.

- After computing the LU factorization:

$$Ax = LUx = b \iff \begin{cases} Lc = b \\ Ux = c \end{cases}$$

where we solve each system by substitution:

(1) Forward substitution:

$$\begin{cases} l_{11} c_1 & & & = b_1 \\ l_{21} c_1 + l_{22} c_2 & & & = b_2 \\ \vdots & & & \\ l_{n1} c_1 + & \cdots & + l_{nn} c_n & = b_n \end{cases}$$

$$\rightarrow \quad c_1 = b_1 / l_{11}$$
$$c_2 = (b_2 - l_{21} c_1) / l_{22}$$
$$\vdots$$
$$c_n = (b_n - l_{n1} c_1 - \cdots - l_{n-1} c_{n-1}) / l_{nn}$$

Cost: $2 \times \frac{n(n-1)}{2} \approx n^2$ operations

(2) Back-substitution: similarly

$$x_i = \frac{1}{u_{ii}}\left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right), \quad j = n, n-1, \ldots, 1.$$

Again, cost is $\sim n^2$.

Existence and Uniqueness.

The k-th leading minor of A is the matrix

$$A_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \\ a_{k1} & \cdots & a_{kk} \end{bmatrix} \quad \text{for } 1 \le k \le n.$$

Theorem. $A \in \mathbb{R}^{n \times n}$ has a unique LU factorization if and only if the minors $A_1, \ldots, A_{n-1}$ are non-singular.

Idea of proof: $A_k = L_k U_k$

and $\det A_k = \det(L_k) \det(U_k)$ ($\det(L_k) = 1$)

$$= \underbrace{u_{11} u_{22} \cdots u_{kk}}_{\text{pivots}} \neq 0.$$

What to do if a pivot is zero?

Ex: $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

2) Pivoting strategies.

a) The need for pivoting.

Consider $A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix}$ (Forsythe & Moler, '67)

in 3-digit arithmetic.

Elimination: multiplier $m_{21} = 1/10^{-4} = 10^4$

$$M_1 = \begin{bmatrix} 1 & 0 \\ -10^4 & 1 \end{bmatrix}, \quad U = M_1 A = \begin{bmatrix} 10^{-4} & 1 \\ 0 & fl(1-10^4) \end{bmatrix}$$

But in 3-digit arithmetic, $1 - 10^4 \approx -10^4$.

Hence the computed factors will be

$$\hat{L} = M_1^{-1} = \begin{bmatrix} 1 & 0 \\ 10^4 & 1 \end{bmatrix}, \quad \hat{U} = \begin{bmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{bmatrix}$$

Now $\hat{L}\hat{U} = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 0 \end{bmatrix}$ and $A - \hat{L}\hat{U} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$!

Large error $\|A - \hat{L}\hat{U}\|_{Fro} = 1$. Problem due to small pivot.

Avoided by exchanging the rows:

$$A' = \begin{bmatrix} 1 & 1 \\ 10^{-4} & 1 \end{bmatrix}, \quad U_1' = \begin{bmatrix} 1 & 0 \\ -10^{-4} & 1 \end{bmatrix}, \quad U' = \begin{bmatrix} 1 & 1 \\ 0 & fl(1-10^{-4}) \end{bmatrix}$$

s.t. $$\hat{L} = \begin{bmatrix} 1 & 0 \\ 10^{-4} & 1 \end{bmatrix}, \quad \hat{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Now $$\hat{L}\hat{U} = \begin{bmatrix} 1 & 1 \\ 10^{-4} & fl(1+10^{-4}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 10^{-4} & 1 \end{bmatrix} = A.$$

b) Permutation matrices.

Definition. A permutation matrix is a square matrix with exactly one non-zero entry in each row and column, equal to one.

If $(\alpha_1, \dots, \alpha_n)$ is a permutation of $(1, \dots, n)$, the associated permutation matrix is

$$P = [e_{\alpha_1} \dots e_{\alpha_n}]^T$$

and $P^{-1} = P^T = [e_{\alpha_1} \dots e_{\alpha_n}]$ is also a permutation matrix.

Effect of multiplying a matrix by P.

Let $P = [e_{\alpha_1} \dots e_{\alpha_n}]^T$ associated with a permutation $(\alpha_1 \dots \alpha_n)$

Then $$PA = \left[\sum_{k=1}^{n} p_{ik} a_{kj}\right]_{ij} = \left[\sum_{k=1}^{n} \delta_{\alpha_i, k} a_{kj}\right] = [a_{\alpha_i, j}]$$

meaning $PA = \begin{bmatrix} \alpha_1^{th}\text{-row of } A \\ \vdots \\ \alpha_n^{th} \text{ row of } A \end{bmatrix}$

i.e. we have permuted the rows of A in order $\alpha_1, \dots, \alpha_n$.

Similarly, $AP^T = \begin{bmatrix} \alpha_1^{th}\text{-col of } A & \cdots & \alpha_n^{th} \text{ col of } A \end{bmatrix}$

is obtained by permuting columns of A in order $\alpha_1 \dots \alpha_n$.

In earlier example, $\alpha = (2,1)$ and $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$,

such that $A' = PA$.

c) Gaussian elimination with partial pivoting.

Here, at each step we look for a good pivot in the k^{th} column of A.

Algorithm. Set $A^{(1)} = A$, then at step k:

(1) identify the largest element (by magnitude) in column k, below the diagonal. Let it be $a^{(k)}_{r_k,k}$ with $r_k \geq k$.

(2) Exchange rows r_k and k, bringing $a^{(k)}_{r_k,k}$ to the diagonal.

(3) Apply the normal k^{th} step of the GEM.

In terms of matrix multiplications, the algorithm yields a sequence:

$$\begin{aligned} A^{(1)} &= A, \\ A^{(2)} &= M_1' P_1 A^{(1)} \\ A^{(3)} &= M_2 P_2 A^{(2)}, \\ &\cdots \\ A^{(n)} &= M_{n-1} P_{n-1} A^{(n-1)}, \end{aligned}$$

where P_k is associated with the permutation of rows k and r_k, i.e.

$$\alpha_i = \begin{cases} r_k, & \text{if } i = k \\ k, & \text{if } i = r_k \\ i, & \text{else} \end{cases}$$

Note that $P_i^T = P_i = P_i^{-1}$.

Then we obtain the upper triangular matrix

$$U = M_{n-1} P_{n-1} \cdots M_1 P_1 A.$$

How to extract L from $M = M_{n-1} P_{n-1} \cdots M_1 P_1$?

Define

$$\begin{cases} M_{n-1}' = M_{n-1}, \\ M_{n-2}' = P_{n-1} M_{n-2} P_{n-1} \\ M_{n-3}' = P_{n-1} P_{n-2} M_{n-3} P_{n-2} P_{n-1} \\ \cdots \\ M_1' = P_{n-1} \cdots P_2 M_1 P_2 \cdots P_{n-1} \end{cases}$$

Then $M = M_{n-1} P_{n-1} M_{n-2} P_{n-2} M_{n-3} P_{n-3} M_{n-4} \cdots M_1 P_1$

$= M_{n-1} (P_{n-1} M_{n-2} P_{n-1}) (P_{n-1} P_{n-2} M_{n-3} P_{n-2} P_{n-1}) (P_{n-1} P_{n-2} P_{n-3} \cdots)$

$\cdots (\cdots M_1 P_2 \cdots P_{n-1}) P_{n-1} \cdots P_2 P_1$

i.e. $M = M'_{n-1} \cdots M'_1 \, (P_{n-1} \cdots P_1)$

Denote $L = (M'_1 \cdots M'_{n-1})^{-1}$ and $P = P_{n-1} \cdots P_1$ then

$$\boxed{PA = LU}$$

Furthermore, construction of the M'_k from the M_k involves left/right multiplications by permutations P_j, $j > k$, permuting rows and columns j and r_j, but

$$M_j = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \\ & & 1 & \\ 0 & & & \ddots \\ & & & & 1 \end{bmatrix}$$

with $-m_{k+1,k}, \ldots, -m_{n,k}$ in column k below the diagonal.

↳ result is to permute the multipliers $-m_{k+1,k}, \ldots, -m_{n,k}$ but keep the structure.

Also by direct computation:

$$\begin{aligned} M'_k &= P_{n-1} \cdots P_{k+1} (I_n - m_k e_k^T) P_{k+1} \cdots P_{n-1} \\ &= I_n - (P_{n-1} \cdots P_{k+1} m_k)(P_{n-1} \cdots P_{k+1} e_k^T) \\ &= I_n - m'_k e_k^T, \end{aligned}$$

where $m'_k = P_{n-1} \cdots P_{k+1} m_k$ is a vector of permuted multipliers.

Then

$$L = (M_1')^{-1} \cdots (M_{n-1}')^{-1} = I_n + m_1' e_1^T + \cdots + m_{n-1}' e_{n-1}^T$$

$$= \begin{bmatrix} 1 & & 0 \\ m_{2,1}' & \ddots & \\ \vdots & & \\ m_{n,1}' & \cdots & m_{n,n-1}' \quad 1 \end{bmatrix}$$

Practical algorithm.

(1) Set $U = A_1$, $L = I_1$, $p = [1 \ \dots \ n]$

(2) For $k = 1, \dots, n-1$:

- Find r_k such that $|u_{r_k,k}| \geq \max_{r \geq k} |u_{rk}|$
- $U[k, k:n] \longleftrightarrow U[r_k, k:n]$ (Exchange rows k, r_k of $A^{(k)}$)
- $L[k, 1:k-1] \longleftrightarrow L[r_k, 1:k-1]$ (Permute multipliers stored in L)
- $p[k] \longleftrightarrow p[r_k]$ (Update permutation p)

- $L[k+1:n, k] = U[k+1:n, k] / U[k,k]$
- $U[k+1:n, k] = 0$
- $U[k+1:n, k+1:n] \mathrel{-}= L[k+1:n, k] \, U[k, k+1:n]$.

End For

(3) Return L, U, p (Now, $A[p, :] = LU$)

d] Complete pivoting.

Here the strategy is to find the largest pivot within the remaining sub matrix:

$$A^{(k)} = \begin{bmatrix} u_{11} & & & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ & & 0 & \\ & & a_{kk}^{(k)} \cdots a_{kn}^{(k)} & \\ & & \vdots \quad\quad \vdots & \\ 0 & \cdots & 0 \; a_{nk}^{(k)} \cdots a_{nn}^{(k)} \end{bmatrix}$$

Assume this is $a_{r_k, s_k}^{(k)}$ with $r_k, s_k \geq k$.

Next we exchange rows k, r_k and columns k, s_k to bring the pivot on the diagonal at (k,k) and perform an elimination step:

$$A^{(k+1)} = M_k P_k A^{(k)} Q_k.$$

At the end of the procedure,

$$U = M_{n-1} P_{n-1} \cdots M_1 P_1 A Q_1 \cdots Q_{n-1}$$

We define $\begin{cases} Q = Q_1 \cdots Q_{n-1}, \quad P = P_{n-1} \cdots P_1, \\ L = (M_1')^{-1} \cdots (M_{n-1}')^{-1} \text{ as before} \end{cases}$

Then $PAQ = LU$ fully pivoted LU.

Practical algorithm.

(1) Set $U = A, \quad L = I, \quad p = [1 \dots n]$

(2) For $k = 1, \dots, n-1$:

- Find r_k, s_k such that $|u_{r_k, s_k}| \geq \max_{r \geq k,\, s \geq k} |u_{r,s}|$
- $U[:, k] \leftrightarrow U[:, s_k]$ (Exchange columns k, s_k of $A^{(k)}$)
- $q[k] \leftrightarrow q[s_k]$ (Update permutation q)

- $U[k, k:n] \leftrightarrow U[r_k, k:n]$ (Exchange rows k, r_k of $A^{(k)}$)
- $L[k, 1:k-1] \leftrightarrow L[r_k, 1:k-1]$ (Permute multipliers stored in L)
- $p[k] \leftrightarrow p[r_k]$ (Update permutation p)

- $L[k+1:n, k] = U[k+1:n, k] / U[k,k]$
- $U[k+1:n, k] = 0$
- $U[k+1:n, k+1:n] \mathrel{-}= L[k+1:n, k]\, U[k, k+1:n]$

End For

(3) Return L, U, p, q (Now, $A[p, q] = LU$)

Cost analysis: additional $\sim n^2/2$ comparisons for GEPP,
$\sim 2n^3/3$ comparisons for GECP
+ exchanges of rows / columns.

February 8

Conditioning of linear Systems.

I) Vector norms.

Definition Let V a vector space over $\mathbb{F}$.
The map $\|\cdot\| : V \to \mathbb{R}^+$ is a norm if

(1) $\|v\| = 0$ implies $v = 0$
(2) $\|\alpha v\| = |\alpha| \|v\|$ for all $\alpha \in \mathbb{F}$, $v \in V$
(3) $\|u + v\| \leq \|u\| + \|v\|$ for all $u, v \in V$.

Examples. • Euclidean norm over $\mathbb{R}^n$:

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} = \sqrt{x^T x}$$

• Hölder p-norm: $\|x\|_p = \left(\sum_{i=1}^{n} x_i^p \right)^{1/p}$, $p \geq 1$

• Infinity-norm: $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$

• Given A full rank, $\|x\|_{A,p} = \|Ax\|_p$

Exercise: check these satisfy (1)–(3).

Definition Two norms $\|\cdot\|$ and $|||\cdot|||$ on V are equivalent if there is $c, C > 0$ such that
$c\|x\| \leq |||x||| \leq C\|x\|$ for all $x \in V$.

Theorem If V is finite-dimensional, all norms on V are equivalent.

Some well-known results:

Proposition. Let $u, v \in \mathbb{F}^n$ with scalar product
$$(u,v) = u^* v = \sum_{i=1}^{n} \bar{u}_i v_i$$

(a) Cauchy-Schwartz inequality:
$$|(u,v)| \leq \|u\|_2 \|v\|_2$$

(b) Hölder inequality: for $p, q \in [1,\infty]$ with $\frac{1}{p} + \frac{1}{q} = 1$,
$$|(u,v)| \leq \|u\|_p \|v\|_q .$$

II] Matrix norm.

Definition A norm on the space of matrices $\mathbb{R}^{m\times n}$ is a vector norm on that space.

A matrix norm is said sub-multiplicative if
$$\|AB\| \leq \|A\| \cdot \|B\| \qquad \forall A \in \mathbb{R}^{m\times n},\ B \in \mathbb{R}^{n\times p}$$

Definition. A matrix norm $\|\cdot\|$ is compatible or consistent with a vector norm, also denoted $\|\cdot\|$ if

$$\|Ax\|_{\mathbb{F}^m} \leq \|A\| \, \|x\|_{\mathbb{F}^n}$$

Not all matrix norms are consistent, or sub-multiplicative

Example: • nuclear norm $\|A\|_\Delta = \max_{i,j} |a_{ij}|$

$$A = B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{then} \quad \|A\|_\Delta = \|B\|_\Delta = 1$$

but $$AB = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \quad \text{so} \quad \|AB\|_\Delta = 2 > \|A\|_\Delta \|B\|_\Delta = 1.$$

• Frobenius norm:

$$\|A\|_F = \left(\sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2} = \sqrt{\operatorname{Tr} A^* A}$$

Property The Frobenius norm is sub-multiplicative and compatible with the Euclidean norm $\|\cdot\|_2$:

also $$\|A\|_F = \|A^T\|_F = \|A^*\|_F.$$

III Induced matrix norms.

Definition. Let $\|\cdot\|$ be a vector norm.

Then $$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|$$

is a matrix norm called induced or natural norm associated with the vector norm $\|\cdot\|$.

Properties If $A \longmapsto \|A\|$ is induced by $\|v\|$,

(1) $\|Ax\| \leq \|A\| \|x\|$ for all $A \in \mathbb{F}^{m\times n}$, $x \in \mathbb{F}^n$

(2) $\|AB\| \leq \|A\| \cdot \|B\|$ for all $A \in \mathbb{F}^{m\times n}$, $B \in \mathbb{F}^{n\times p}$

(3) $\|I_n\| = 1$.

Example 1. Case $\|v\|_1 = \sum_{i=1}^{n} |v_i|$

Here $$\|Ax\|_1 = \Big\| \sum_{j=1}^{n} x_j c_j \Big\|_1 \leq \sum_{j=1}^{n} |x_j| \|c_j\|_1$$

(c_j: columns of A)

so $$\|Ax\|_1 \leq \underbrace{\Big(\max_{1\leq j\leq n} \|c_j\|_1 \Big)}_{C} \underbrace{\sum_{j=1}^{n} |x_j|}_{\|x\|_1}$$

This shows $$\frac{\|Ax\|_1}{\|x\|_1} \leq C \Rightarrow \|A\|_1 \leq C.$$

On the other hand, let j maximizing $\|c_j\|_1$ and $x = e_j$ then

$$\frac{\|Ac_j\|}{\|e_j\|} = \frac{\|c_j\|_1}{1} = C \leq \|A\|_1$$

February 10

Conditioning of linear systems

Definition. Let $\|\cdot\|$ be a vector norm.

Then $$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|$$

Example 1. Case $\|v\|_1 = \sum_{i=1}^{n} |v_i|$

Here $$\|Ax\|_1 = \Big\| \sum_{j=1}^{n} x_j c_j \Big\|_1 \leq \sum_{j=1}^{n} |x_j| \, \|c_j\|_1$$

(c_j: columns of A)

so $$\|Ax\|_1 \leq \underbrace{\Big(\max_{1 \leq j \leq n} \|c_j\|_1 \Big)}_{C} \underbrace{\sum_{j=1}^{n} |x_j|}_{\|x\|_1}$$

This shows $$\frac{\|Ax\|_1}{\|x\|_1} \leq C \Rightarrow \|A\|_1 \leq C.$$

On the other hand, let j maximizing $\|c_j\|_1$ and $x = e_j$ then

$$\frac{\|Ac_j\|}{\|e_j\|} = \frac{\|c_j\|_1}{1} = C \leq \|A\|_1$$

This shows $$\|A\|_1 = \max_{j=1,\dots,n} \|c_j\|_1 = \max_{j=1\dots n} \Big(\sum_{i=1}^{m} |a_{ij}| \Big)$$

"column sum norm"

Example 2. Case $p = \infty$. Here one can show

$$\|A\|_\infty = \max_{i=1\dots n} |r_i|_1 = \max_{i=1\dots m} \Big(\sum_{j=1}^{n} |a_{ij}| \Big)$$

"row sum norm"

Example 3. Case $p = 2$.

Note $\|A\|_2^2 = \max_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \max_{x \neq 0} \frac{x^T A^T A x}{x^T x}$.

Now A^TA is symmetric and positive (i.e. $x^T(A^TA)x \geq 0 \ \forall x$) so it is diagonalizable with orthogonal eigenvectors and positive eigenvalues $\sigma_1^2 \geq \sigma_2^2 \geq \dots \geq \sigma_n^2$

↳ if $x = \sum_{i=1}^{n} x_i v_i$, $x^TA^TAx = \sum_{i=1}^{n} \sigma_i^2 x_i^2 \leq \sigma_1^2 \sum_{i=1}^{n} x_i^2 = \sigma_1^2 x^T x$ with equality for $x = v_1$

Hence $\|A\|_2 = \sigma_1$ where σ_1^2 is the largest eigenvalue of A^TA, also called the first singular value of A.

Proposition Let $A \in \mathbb{F}^{m \times n}$ then

(1) $\|A\|_1 = \|A^T\|_\infty = \|A^*\|_\infty$

(2) $\|A\|_\infty = \|A^T\|_1 = \|A^*\|_1$

(3) $\|A\|_2 \leq \|A\|_F$

(4) $\|A\|_2^2 \leq \|A\|_1 \|A\|_\infty$

IV Stability Analysis.

Consider $Ax = b$.
How sensitive is the solution x to perturbations in A, b?

Absolute vs Relative error: Given $y \approx x$ we measure

- Absolute error: $\|x - y\|$
- Relative error: $\frac{\|x-y\|}{\|x\|}$

Example: $x = 10{,}000$ and $y = 10{,}001$
$x = 0{,}001$ and $y = 1{,}001$

Definition **Condition number of a matrix.**

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

where A is an induced norm. If A is not invertible, we set $\kappa(A) = +\infty$

Remarks

- Depends on norm: denote $\kappa_\infty(A)$, $\kappa_2(A)$...
- $\kappa(A) \geq 1$ since $\|A\|\|A^{-1}\| \geq \|AA^{-1}\| = \|I\| = 1$.
- $\kappa(A) = \kappa(A^{-1})$

Applications.

Mat-vec $x = Ab$ for inexact b:

$$x + \delta x = A(b + \delta b)$$
$$\rightarrow \quad \delta x = A\,\delta b$$
$$\|\delta x\| \leq \|A\|\,\|\delta b\|$$

On the other hand $\|b\| = \|A^{-1}x\| \leq \|A^{-1}\| \cdot \|x\|$

hence $$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A\|\,\|\delta b\|}{\|b\| / \|A^{-1}\|} \leq \kappa(A)\frac{\|\delta b\|}{\|b\|}$$

Solution of linear system: $Ax = b$ vs $A(x + \delta x) = b + \delta b$

Assuming perfect application of A^{-1}: same as computing $b = A^{-1}x$

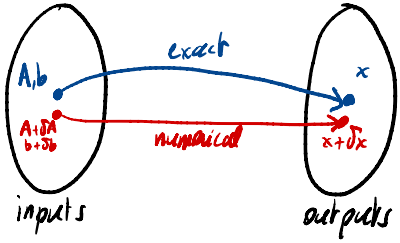
Since $\kappa(A^{-1}) = \kappa(A)$:

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\delta b\|}{\|b\|}$$

IV Three kinds of stability analysis.

A) Forward analysis

Idea : process is inexact → perturbed inputs



$\Rightarrow \quad \frac{\|\delta x\|}{\|x\|}$?

Theorem. Let $A, \delta A$ such that $\|A^{-1}\| \, \|\delta A\| < 1$,
and $x, \delta x, b, \delta b$ such that $b \neq 0$,
$Ax = b$ and $(A - \delta A)(x + \delta x) = b + \delta b$

Then $$\frac{\|\delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right)$$

Idea of proof. By linearity:

$$A \, \delta x - \delta A (x + \delta x) = \delta b$$

or $$\delta x = (A - \delta A)^{-1} [\, \delta b + \delta A \, x]$$

Let $S = (I - A^{-1} \delta A)^{-1} = (A - \delta A)^{-1} A$: $\quad \delta x = S A^{-1} [\, \delta b + \delta A \, x]$

Then $S (I - A^{-1} \delta A) = I$ hence $S = I + S A^{-1} \delta A$

$$\|S\| \leq 1 + \|S\| \, \|A^{-1}\| \, \|\delta A\|$$

$$\Rightarrow \quad \|S\| \leq \frac{1}{1 - \|A^{-1}\| \, \|\delta A\|}$$

Thus $$\frac{\|\delta x\|}{\|x\|} \le \frac{\|A^{-1}\|}{1 - \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}} \left[\frac{\|\delta b\|}{\|x\|} + \|\delta A\| \|x\| \right]$$

Since $\|b\| \le \|A\| \|x\|$,

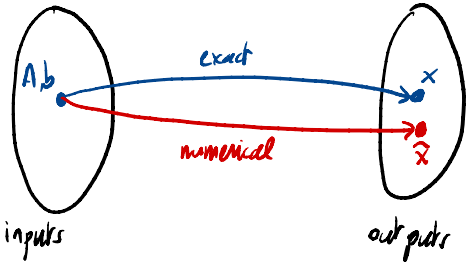
$$\frac{\|\delta x\|}{\|x\|} \le \frac{\|A^{-1}\| \|A\|}{1 - \kappa(A)\frac{\|\delta A\|}{\|A\|}} \left[\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right] \quad \square$$

Particularly, if $\frac{\|\delta A\|}{\|A\|} \approx \frac{\|\delta b\|}{\|b\|} \approx \varepsilon$ (machine precision: 10^{-16})

we find $$\frac{\|\delta x\|}{\|x\|} \le \frac{2\kappa(A)}{1 - \varepsilon\kappa(A)} \varepsilon .$$

B) Backwards analysis.

New frame of mind: numerical algorithm produces exact solution to approximate problem: $\hat{x} = Cb$ where $C \approx A^{-1}$.



We see C as the exact inverse / solution operator to a modified problem with perturbed matrix $A + \delta A$, with

$$\delta A = C^{-1} - A = (I - AC)C^{-1} = -RC^{-1}$$

where $R = AC - I$ should be small:

$$\|\delta A\| \leq \|R\| \, \|C^{-1}\|,$$

and since $RC^{-1} + C^{-1} = A$,

$$\|C^{-1}\| \leq \|A - RC^{-1}\| \leq \|A\| + \|R\| \, \|C^{-1}\|$$

so $$\|C^{-1}\| \leq \frac{\|A\|}{1 - \|R\|}$$

and $$\|\delta A\| \leq \frac{\|A\| \, \|R\|}{1 - \|R\|}.$$

c) A-posteriori analysis.

Finally: given an approximate solution $y \simeq x = A^{-1}b$

one may seek to estimate the error $e = y - x$ from known quantities, including y.

We may start from the residual $r = b - Ay$,

since $e = A^{-1}(Ay - b) = -A^{-1}r$,

so $$\|e\| \leq \frac{\|C\|}{1 - \|R\|} \|r\|.$$

Alternatively since $Ay = b + r$, set $r = \delta b$ and we obtain

$$\frac{\|e\|}{\|x\|} \leq \kappa(A) \frac{\|r\|}{\|b\|}.$$

Such formulae (also including rounding error effects) are in use in modern linear algebra libraries such as LAPACK.

February 15

Solution of linear Systems Stability.

I] Practical solution of linear systems.

$$Ax = b. \qquad A = \begin{bmatrix} \varepsilon & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad 0 \leq \varepsilon \ll 1$$

① Compute the LU factorization.
non pivoted

Step 1 Multiplies $m_1 = [0, 1/\varepsilon, 0]$

$$M_1 = I_3 - m_1 e_1^T = \begin{bmatrix} 1 & 0 & 0 \\ -1/\varepsilon & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad A^{(1)} = \begin{bmatrix} \varepsilon & 1 & 0 \\ 0 & 1-1/\varepsilon & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Step 2 $m_2 = \left[0, 0, \frac{\varepsilon}{\varepsilon-1}\right]$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\varepsilon/\varepsilon-1 & 1 \end{bmatrix} \qquad U = A^{(2)} = \begin{bmatrix} \varepsilon & 1 & 0 \\ 0 & \frac{\varepsilon-1}{\varepsilon} & 1 \\ 0 & 0 & 1-\varepsilon/\varepsilon-1 \end{bmatrix}$$

$$L = M_1^{-1} M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1/\varepsilon & 1 & 0 \\ 0 & \varepsilon/\varepsilon-1 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} \varepsilon & 1 & 0 \\ 0 & \varepsilon-1/\varepsilon & 1 \\ 0 & 0 & 1/1-\varepsilon \end{bmatrix}$$

Check $A = LU$.

To solve $Ax = b$, solve $Lc = b$ then $Ux = c$

Example: $Ax = e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/\varepsilon & 1 & 0 \\ 0 & \varepsilon/\varepsilon-1 & 1 \end{bmatrix} c = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} c_1 = 1 \\ c_2 = 0 - 1/\varepsilon \, c_1 = -1/\varepsilon \\ c_3 = 0 - \frac{\varepsilon}{\varepsilon-1} c_2 = -1/\varepsilon-1 \end{cases}$$

$$\begin{bmatrix} \varepsilon & 1 & 0 \\ 0 & \varepsilon-1/\varepsilon & 1 \\ 0 & 0 & 1/1-\varepsilon \end{bmatrix} x = \begin{bmatrix} 1 \\ -1/\varepsilon \\ 1/\varepsilon-1 \end{bmatrix} \Rightarrow \begin{cases} \varepsilon x_1 = 1 - x_2 \\ \frac{\varepsilon-1}{\varepsilon} x_2 = -1/\varepsilon - x_3 \\ \frac{\varepsilon}{1-\varepsilon} x_3 = \frac{1}{\varepsilon-1} \end{cases} \quad \begin{cases} x_1 = 0 \\ x_2 = 1 \\ x_3 = -1 \end{cases}$$

Hence $x = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} = A^{-1} e_1$ First column of A^{-1} !

Similarly solve $Ax = e_2$, $Ax = e_3$:

$$A^{-1} = \begin{bmatrix} 0 & 1 & -1 \\ 1 & -\varepsilon & \varepsilon \\ -1 & \varepsilon & 1-\varepsilon \end{bmatrix}$$

Conditionning. We find $\kappa_1(A) = \|A\|_1 \, \|A^{-1}\|_1$

Recall $\|\cdot\|_1$ is the column sum norm:

$$\|A\|_1 = \max\{1+\varepsilon, 3, 2\} = 3$$
$$\|A^{-1}\|_1 = \max\{2, 1+2\varepsilon, 2\} = 2$$

$\kappa(A) = 6$

By solving using LU, we have $\frac{\|\delta x\|}{\|x\|} \leq \kappa(U)\kappa(L) \frac{\|\delta b\|}{\|b\|}$

But we have $\kappa(U) \sim \kappa(L) \sim 1/\varepsilon^2$! Terrible stability
For $\varepsilon \sim 10^{-4}$ floating-point errors result in 100% errors.

Partial pivoting

$$A = \begin{bmatrix} \varepsilon & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Step 1. Pick $a_{21} = 1$ as pivot: $1 \leftrightarrow 2$

$$P_1 A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad m_1 = \begin{bmatrix} 0 & \frac{\varepsilon}{1} & \frac{0}{1} \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -\varepsilon & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \longrightarrow A^{(1)} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1-\varepsilon & -\varepsilon \\ 0 & 1 & 1 \end{bmatrix}$$

At this point $p = [2, 1, 3]$ $\quad L = \begin{bmatrix} 1 & 0 & 0 \\ \varepsilon & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\quad U = \begin{bmatrix} 1 & \varepsilon & 1 \\ 0 & 1-\varepsilon & -\varepsilon \\ 0 & 1 & 1 \end{bmatrix}$

Step 2. Pick $u_{32} = 1$ as new pivot:

$$P_2 U = \begin{bmatrix} 1 & \varepsilon & 1 \\ 0 & 1 & 1 \\ 0 & 1-\varepsilon & -\varepsilon \end{bmatrix} \qquad m_2 = \begin{bmatrix} 0 & 0 & \frac{1-\varepsilon}{1} \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \varepsilon-1 & 1 \end{bmatrix} \Rightarrow A^{(2)} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

Now $p = [2, 3, 1]$ $\quad L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \varepsilon & 1-\varepsilon & 1 \end{bmatrix}$ $\quad U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$

We have $$PA = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ \varepsilon & 1 & 0 \end{bmatrix} = LU.$$

However this time $\kappa_1(P) = 1$, $\kappa_1(L) \sim (2-\varepsilon)^2$ and $\kappa_1(U) = 6$

Slightly larger condition number but not much!

II) Error estimates for Gaussian elimination.

When taking into account rounding errors (floating-point), GEM produces factors $\hat{L}, \hat{U}$ such that

$$\hat{L}\hat{U} = A + \delta A$$

↑ Results from imperfect computation.

Goal: stability bound $\|\delta A\| \leq \rho(A) \|A\|$

need to control growth of elements during elimination process:

$$A^{(1)}, A^{(2)} \dots A^{(n-1)} = U.$$

Def. Growth factor $\rho(A) = \dfrac{\max \alpha_1, \dots, \alpha_{n-1}}{\alpha}$

where $\alpha = \max_{i,j} |a_{ij}|$ and $\alpha_k = \max_{i,j} |a_{ij}^{(k)}|$

Note: when pivoting is used, elements of L are bounded:

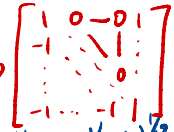
$$|\ell_{ik}| = \left| \frac{a_{ik}^{(k)}}{a_{r_k,k}^{(k)}} \right| \leq 1$$

since the pivot is the largest entry by magnitude in column, or even larger.

Furthermore $|u_{ij}| \leq \rho \max_{ij} |a_{ij}|$.

Theorem $\hat{L}\hat{U} = A + \delta A$ where
$\|\delta A\|_\infty \leq 8 n^3 \rho(A) \|A\|_\infty + O(u^2)$
with u the machine precision.

How big can the growth factor become?
- With partial pivoting: $\rho(A) \leq 2^{n-1}$
- With complete pivoting: $\rho(A) \leq \sqrt{n}\left(2^1 3^{1/2} 4^{1/3} \cdots n^{1/n-1}\right)^{1/2}$
- No pivoting: unbounded. Completely unstable.



Iterative Refinement. Assume we compute an approximate solution $\hat{x}$ to $Ax = b$, using e.g. an LU decomposition. Usually the residual $r = b - A\hat{x}$ is nonzero but small. Then we can solve again

$$A\hat{c} \approx r$$

and because $\|r\| \ll \|b\|$, the error on c is much smaller than that on x. Hence $\hat{y} = \hat{x} + \hat{c}$ better approximation to x.

Given $A, b, \hat{x}^{(0)}$
For $i = 0, 1, \ldots$ until convergence:
- Compute $r^{(i)} = b - A\hat{x}^{(i)}$
- Solve $Az = r^{(i)}$
- Update $\hat{x}^{(i+1)} = \hat{x}^{(i)} + z$
- If $\|z\| / \|x^{(i+1)}\| <$ tolerance: return $\hat{x}^{(i+1)}$

End.

II) Other decompositions.

If the leading principal minors $A_1, \dots, A_n$ are non-singular write $A = LU$ (no pivoting)

Then $U = D \cdot M^T$ with $M = (D^{-1}U)^T$ unit lower triangular (i.e. $m_{ii} = 1$)

(D = diagonal of U)

Hence $\boxed{A = LDM^T}$

with D diagonal, L, M unit lower triangular.

If A is symmetric then $A^T = MDL^T$ so $L = M$ by uniqueness:

$$\boxed{A = LDL^T}$$

If A is symmetric positive definite: $x^T A x > 0$ for $x \neq 0$, so are its leading minors $A_1, \dots, A_n$: $x_k A_k x_k = \begin{bmatrix} x_k \\ 0 \end{bmatrix}^T A \begin{bmatrix} x_k \\ 0 \end{bmatrix} > 0$.

Hence $\det(A_k) = u_{11} \cdots u_{kk} > 0$.

Then $D = \begin{bmatrix} u_{11} & & 0 \\ & \ddots & \\ 0 & & u_{nn} \end{bmatrix} = \begin{bmatrix} \sqrt{u_{11}} & & 0 \\ & \ddots & \\ 0 & & \sqrt{u_{nn}} \end{bmatrix}^2 = S^2$

hence $A = LS^2L^T = (LS)(LS)^T = HH^T$

with H lower triangular with positive diagonal entries.

→ **Cholesky factorization:** $\boxed{A = HH^T}$

Cheap ($\sim n^3/3$) and stable for SPD matrices.

III Orthogonality.

① **Orthogonal vectors:** $x \cdot y = x^T y = 0$

(complex: $x^* y = \sum_{i=1}^{n} \overline{x_i} y_i = 0$)

② **Orthogonal family:**

nonzero vectors $v_1, \dots, v_m$ with $v_i \cdot v_j = 0$ for $i \neq j$

ensures independent!

Orthonormal basis: unit vectors $v_1, \dots, v_n \in \mathbb{R}^n$ with

$$v_i \cdot v_j = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

each element has norm 1.

③ **Orthogonal subspaces:** $R, N \subset V$ are orthogonal if $x \cdot y = 0$ for all $x \in R, y \in N$.

Example: • $R = \text{Span}\{v_1, \dots, v_k\}$ $N = \text{Span}\{v_{k+1}, \dots, v_m\}$

where $v_1, \dots, v_m$ orthogonal family and $1 \leq k < m$

• $R = \text{Ran } A^T$ and $N = \text{Ker } A$

row space of A kernel or null-space

since $\begin{bmatrix} \text{row 1 of } A \\ \vdots \\ \text{row m of } A \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

④ Matrix with orthonormal columns: for $m \leq n$

$$Q^TQ = I = \begin{bmatrix} q_1^T \\ \vdots \\ q_n^T \end{bmatrix} \begin{bmatrix} q_1 \cdots q_n \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

When such a matrix multiplies a vector x, length is conserved:

$$\|Qx\|_2 = \sqrt{(Qx)^TQx} = \sqrt{x^TQ^TQx} = \sqrt{x^Tx} = \|x\|_2$$

⑤ Orthogonal matrices: square, columns form an orthonormal basis

$Q^TQ = I$ or $Q^T = Q^{-1}$ hence $QQ^T = I$ for free.

$\rightsquigarrow$ the rows of Q are another orthonormal basis.

Examples. ① $x = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix}$

Note $\|x-y\|^2 = \|x\|^2 + \|y\|^2 - 2\|x\|\|y\|\cos\theta$
where θ angle between x, y. If orthogonal: Pythagoras
$\|x-y\|^2 = \|x\|^2 + \|y\|^2$.

②③ Standard basis $\vec{i} = e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\vec{j} = e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\vec{k} = e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

④⑤ Hadamard matrices.

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix}$$

Thursday, Feb 24 — Least-Squares

I Geometry of matrices with orthogonal columns

Take

$$Q_1 = \frac{1}{3}\begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} \qquad Q_2 = \frac{1}{3}\begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \qquad Q_3 = \frac{1}{3}\begin{bmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 2 & 2 \end{bmatrix}$$

Each of these satisfies $Q_n^T Q_n = I_n$.

Notice $P_n = Q_n Q_n^T$ has rank n (not the identity unless $n = 3$!)

has the special property

$$P_n^2 = (Q_n Q_n^T)(Q_n Q_n^T) = Q_n \overbrace{(Q_n^T Q_n)}^{=I_n} Q_n^T = Q_n Q_n^T = P_n$$

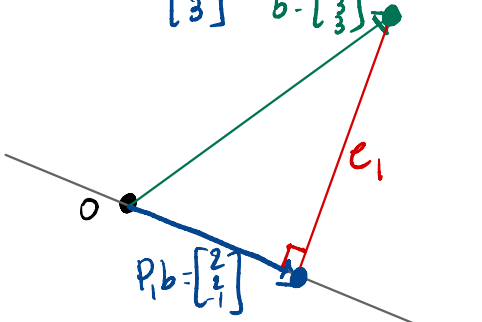
indicating a projection matrix.

Indeed $P = P^2 = P^T$ indicates an orthogonal projection onto the column space $\mathrm{Ran}\, P$.

Ex: $b = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$

$$P_1 b = \frac{1}{9}\begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}\begin{bmatrix} 2 & 2 & -1 \end{bmatrix}\begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

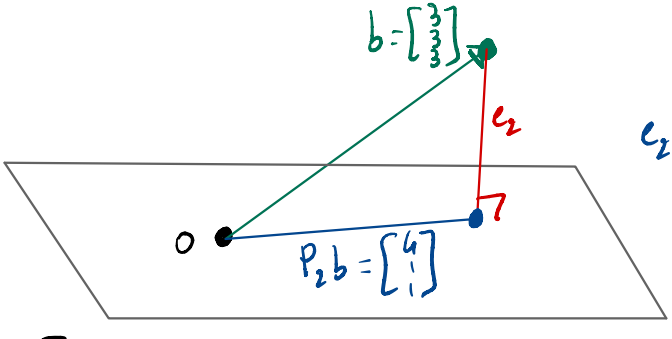
$$= \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$ projection on line



Note the error $e_1 = b - P_1 b = \begin{bmatrix} 1 \\ 1 \\ 4 \end{bmatrix}$, $\|e_1\| = \sqrt{18}$

Next $P_2 b = \frac{1}{9} \begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 & -1 \\ 2 & -1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$

$= \frac{1}{9} \begin{bmatrix} 2 & 2 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 9 \\ 9 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix}$



$e_2 = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$ $\|e_2\| = \sqrt{9}$

better approximation!

Finally: $P_3 = I$ so $P_3 b = b$, $\|e_3\| = 0$.

$\Rightarrow$ $P_n b$ best approximation of b in $\mathrm{Ran}\, b$.

LEAST SQUARES!

II) Orthogonal matrices.

If Q square: $Q^T Q = Q Q^T = I$. $Q^T = Q^{-1}$.

Very important transformations: conserve angles and distances.

$\det Q^T Q = (\det Q^T)(\det Q) = |\det Q|^2 = 1$

Hence $|\det Q| = 1$! Real case: $\det Q = \pm 1$.

Orthogonal matrices form a **group**: Q_1, Q_2 orthogonal then

$$(Q_1 Q_2)^T (Q_1 Q_2) = Q_2^T (Q_1^T Q_1) Q_2 = Q_2^T Q_2 = I.$$

- Orthogonal group $O(n) = \{ Q \in \mathbb{R}^{n \times n} \text{ orthogonal} \}$
- Special orthogonal $SO(n) = \{ Q \in O(n), \det Q = 1 \}$

An orthogonal matrix encodes coordinate transformation between two orthogonal basis.

If $Q = [q_1 \dots q_n]$ is unitary,

then $q_1 \dots q_n$ form a basis of $\mathbb{R}^n$ so for any vector $v \in \mathbb{R}^n$,

$$v_1 e_1 + \dots + v_n e_n = v = c_1 q_1 + \dots + c_n q_n$$

The coefficients are easily calculated:

$$c_1 = q_1 \cdot v = q_1^T v, \dots, c_n = q_n^T v$$

or $$c = Q^T v,$$

and $$\sum_{i=1}^{n} c_i^2 = \|v\|_2^2 = \sum_{i=1}^{n} v_i^2$$

Examples: in dimension $n=2$,

- **Rotations:** $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

(note $\det R_\theta = 1$)

- **Reflections:** $S_\theta = R_\theta \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} R_{-\theta} = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix}$

Reflection across θ-line. ($\det S_\theta = -1$)

- **Householder reflections.** Select $u \in \mathbb{R}^n$ unit vector: $\|u\|_2 = 1$.

$$\boxed{H = I_n - 2uu^T \text{ is a Householder reflector.}}$$

Note H is symmetric: $H = H^T$, and

$$H^T H = (I - 2uu^T)(I - 2uu^T) = I - 4uu^T + 4u(u^T u)u^T = I_n \text{ if } u^T u = 1.$$

Key property: $Hu = -u$
and $Hv = v$ if $u \perp v$.

Example: $u = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $2uu^T = \frac{2}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$$H_3 = \frac{1}{3}\begin{bmatrix} 1 & -2 & -2 \\ -2 & 1 & -2 \\ -2 & -2 & 1 \end{bmatrix}$$

Math 1 Four ways to solve the least-squares problem

Recall: we want to solve an unsolvable problem

$$Ax \approx b$$

in the following way:

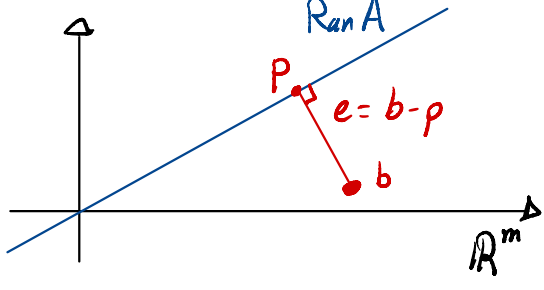
Find $\hat{x}$ minimizing $\|Ax-b\|_2^2$ (1)

Here $A \in \mathbb{R}^{m \times n}$.

A) Normal equations.

Assume here rank $A = n$: the columns of A are independent.

Recall the picture



→ Usually b not in column space; best vector has $A\hat{x} = p$, projection on Ran A.

Clearly $e = b-p \perp$ Ran A

i.e. $A^T(b - A\hat{x}) = 0$

which forces the normal equations:

$$A^T A \hat{x} = A^T b \quad (2)$$

Now we note that A and $A^T A$ share the same null-space! indeed

$$\|Ax\|_2^2 = (Ax)^T Ax = x^T A^T Ax$$

hence $A^T Ax = 0 \Rightarrow Ax = 0.$

Since $Ax = 0 \Rightarrow A^T Ax = 0,$

clearly $\operatorname{Ker} A^T A = \operatorname{Ker} A.$

In particular, if $\operatorname{rank} A = n$, $\operatorname{Ker} A = \{0\}$

so $A^T A \in \mathbb{R}^{n \times n}$ is invertible.

Hence we may solve LS problem (1) by solving (2).

$$\hat{x} = (A^T A)^{-1} A^T b.$$

Remarks. ① $A^T A$ is nice: indeed it is SPD:

$(A^T A)^T = A^T A$, $\quad x^T (A^T A) x = \|Ax\|_2^2 > 0$ for $x \neq 0.$

② Formula for projection onto $\operatorname{Ran} A$:

$$p = A\hat{x} = A(A^T A)^{-1} A^T b$$

$\rightarrow$ Projection matrix: $P = A(A^T A)^{-1} A^T.$

③ Bad idea in practice: condition number squared.

B] Gram-Schmidt process.

To create a set of orthogonal vectors out of some independent collection $v_1, \dots, v_m$.

Step 1 Normalize $q_1 = v_1 / \|v_1\|_2$

Step 2 Orthogonalize $p_2 = v_2 - (q_1^T v_2) q_1$ $= (q_1 q_1^T) v_2$

Normalize $q_2 = p_2 / \|p_2\|$

Notice that $\{q_1, q_2\}$ form an orthonormal family.

Indeed $q_1^T p_2 = q_1^T v_2 - (q_1^T v_2)(q_1^T q_1) = 0$

We continue:

Step 3 O: $p_3 = v_3 - \underbrace{\left[(q_1^T v_3) q_1 + (q_2^T v_3) q_2\right]}_{Q_2 Q_2^T v_3 \text{ with } Q_2 = [q_1\ q_2]}$

N: $q_3 = p_3 / \|p_3\|$

...

Step k: Orth: $p_{k+1} = v_{k+1} - \underbrace{\left[(q_1^T v_{k+1}) q_1 + \dots + (q_k^T v_{k+1}) q_k\right]}_{\text{Projection on Span}\{q_1, \dots, q_k\}\ Q_k Q_k^T v_{k+1}}$

Norm: $q_{k+1} = p_{k+1} / \|p_{k+1}\|$

↳ This process produces orthonormal q's from independent vectors a's.

Let us apply this algorithm to columns of A:

$q_1, \dots, q_n$ from $a_1, \dots, a_n \in \mathbb{R}^m$, $m \geq n$

$$q_1 = a_1 / \|a_1\|$$

$$q_k = \frac{1}{\|p_k\|} \underbrace{\left[a_k - (q_1^T a_k) q_1 - \dots - (q_{k-1}^T a_k) q_{k-1} \right]}_{p_k}$$

a's from q's:

$$\begin{cases} a_1 = \|p_1\| q_1 \\ a_2 = (q_1^T a_2) q_1 + \|p_2\| q_2 \\ \vdots \\ a_n = (q_1^T a_n) q_1 + \dots + (q_{n-1}^T a_{n-1}) q_{n-1} + \|p_n\| q_n \end{cases}$$

Another way of formulating this:

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} = \underbrace{\begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \|p_1\| & q_1^T a_1 & \cdots & q_1^T a_n \\ & \|p_2\| & \ddots & q_{n-1}^T a_n \\ & & & \|p_n\| \end{bmatrix}}_{R}$$

$\Rightarrow \quad A = QR$

orthogonal — upper triangular

Now, $Q \in \mathbb{R}^{m \times n}$ and

$\text{Ran } Q = \text{Span}\{q_1, \dots, q_n\} = \text{Span}\{a_1, \dots, a_n\} = \text{Ran } A$

and Q orthonormal columns : $Q^T Q = I$.

↳ Projector on $\text{Ran } A$: $P = QQ^T$

↳ $p = A\hat{x} = QQ^T b$

multiply by Q^T on both sides :

$$Q^T A \hat{x} = Q^T Q R \hat{x} = Q^T Q b$$

or $Rx = Q^T b$

Note that in terms of maps, we have

$$x \longmapsto Rx \longmapsto QRx = p$$

$$x \xrightarrow{\quad A \quad} p$$

$\rightsquigarrow$ R is the representation of A in the orthonormal basis $q_1, \dots, q_n$ of $\text{Ran } A$!

Conclusion. with some additional twists (column pivoting, Householder algorithm) cheap and stable way to solve least-squares problem.

c) Rank-deficient case. What if $\operatorname{rank} A < n$?

→ Now $\dim \ker A = n - \operatorname{rank} A > 0$

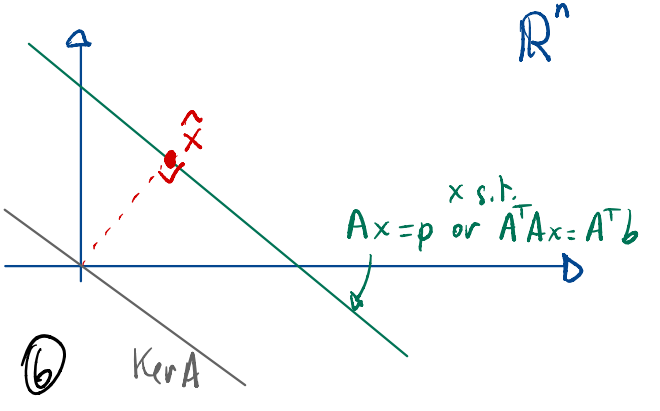
↳ many solutions to $Ax = p$ where p orthogonal projection on $\operatorname{Range}(A)$.

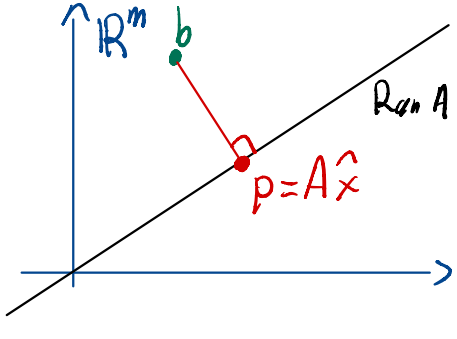
In particular, A^TA is not invertible any more.

Main idea: we seek the solution $\hat{x}$ with smallest norm

$$\hat{x} = \operatorname{argmin} \|x\|^2 \quad \text{subject to} \quad A^TAx = A^Tb.$$

still valid condition: $Ax - b \perp \operatorname{Ran} A$





Tikhonov regularization. For small $\delta > 0$, minimize instead: $\|Ax - b\|_2^2 + \delta^2 \|x\|_2^2$ (*)

↳ As $\delta \to 0$, satisfies $\hat{x}_\delta \to \hat{x}$ minimum norm solution.

How to find an equation for the minimum?

↳ calculus: find where gradient vanishes!

$$F(x+h) = F(x) + \nabla f(x)^T h + \mathcal{O}(h^2)$$

$$\|A(x+h) - b\|_2^2 = \big(A(x+h) - b\big)^T \big(A(x+h) - b\big)$$

$$= (Ax-b)^T(Ax-b) + (Ah)^T(Ax-b) + (Ax-b)^T Ah + (Ah)^T Ah$$

$$= \|Ax-b\|_2^2 + \underbrace{2(A^TAx - A^Tb)^T h}_{\text{order 1 term: gradient}} + \|Ah\|_2^2$$

$$\|x+h\|_2^2 = (x+h)^T(x+h) = \|x\|_2^2 + 2x^Th + \|h\|_2^2$$

↳ Gradient of (*) vanishes if

$$A^TAx - A^Tb + \delta^2 x = 0$$

or $\boxed{(A^TA + \delta^2 I)x = A^Tb}$

Regularized normal equations

$\delta \to 0$

Always well posed for $\delta > 0$.

D) Pseudo-inverse and SVD

We define the pseudo inverse A^+ as the best solution:

$\hat{x} = A^+ b$

as defined through A | B | C above.

$\Rightarrow$ $A \in \mathbb{R}^{m \times n}$ then $A^+ \in \mathbb{R}^{n \times m}$.

Example: $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ then $A^+ = \begin{bmatrix} 1/2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

Fundamental relations:

- If b in column space $\mathrm{Ran}\, A$,

$$AA^+b = b.$$

- If b in row space $\mathrm{Ran}\, A^T = (\mathrm{Ker}\, A)^\perp$,

$$A^+Ax = x.$$

↳ A^+ inverts A whenever possible.

⇒ If A has rank k, let

* $U = [u_1 \dots u_k] \in \mathbb{R}^{m \times k}$ orthonormal basis of $\operatorname{Ran} A$
* $V = [v_1 \dots v_k] \in \mathbb{R}^{n \times k}$ orthonormal basis of $\operatorname{Ran} A^T$

Now $S = U^T A V \in \mathbb{R}^{k \times k}$ has rank k ⇒ invertible.

Note $A = UU^T A VV^T = USV^T$

UU^T: projector on $\operatorname{Ran} A$; VV^T: projector on $\operatorname{Ran} A^T$

Then $A^+ = VS^{-1}U^T$

How to implement this?

Theorem (SVD) Let $A \in \mathbb{R}^{m \times n}$. Then there exists orthogonal matrices $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ such that

$$A = U \Sigma V^T$$

where $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m \times n}$, $p = \min(m,n)$

$$\sigma_1 \geq \dots \geq \sigma_p \geq 0.$$

Case	$m < n$	$m = n$	$m > n$
	$\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{bmatrix}$	$\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$	$\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix}$

* $\sigma_1, \dots, \sigma_p$ are the singular values of A.

* Columns of V form the right singular vectors:

$$A v_j = U \Sigma (V^T v_j) = U \Sigma e_j = \sigma_j u_j$$

* Columns of U form the left singular vectors:

$$u_i^T A = (u_i^T U) \Sigma V^T = e_i^T \Sigma V^T = \sigma_i v_i^T$$

* $AA^T = U \Sigma \Sigma^T U^T$ and $A^T A = V \Sigma^T \Sigma V^T$

$m \times m$ diagonal; $n \times n$ diagonal

$\operatorname{diag}(\sigma_1^2, \dots, \sigma_p^2, 0, \dots, 0)$

$m-p$ zeros if $m > n$

* $k = \operatorname{rank} A = \operatorname{rank} \Sigma = \#$ non zero σ_i's

Then $u_1, \dots, u_k$ form an orthonormal basis of $\operatorname{Ran} A$

$v_1, \dots, v_k$ form an orthonormal basis of $\operatorname{Ran} A^T$

$$A = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix}$$

Tuesday, March 7

Geometry of the SVD. Applications.

I) SVD and Fundamental Subspaces.

Recall the SVD of a real matrix of rank k:

$$A = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i \underbrace{u_i v_i^*}_{m\times n,\ \text{rank-1}} = U_r \Sigma_r V_r^T$$

U: $m\times m$, Σ: $m\times n$, V^T: $n\times n$ — Full SVD

Dyadic SVD

U_r: $m\times r$, Σ_r: $r\times r$, V_r^T: $n\times r$ — Reduced SVD

Note $\sigma_1 \geq \dots \geq \sigma_r > 0$, $\sigma_{r+1} = \dots = \sigma_p = 0$, $p = \min(m,n)$.

The SVD has these properties:

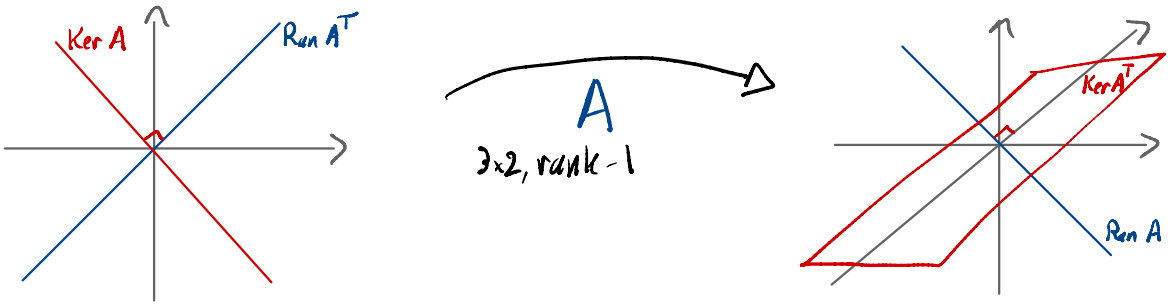
Theorem: Fundamental Theorem of Linear Algebra.
Suppose $A \in \mathbb{R}^{m\times n}$, rank $A = r$, with full SVD $A = U\Sigma V^*$.

(1) Column space: $\operatorname{Ran} A = \operatorname{range} U_r = \operatorname{Span}\{u_1, \dots, u_r\} \subset \mathbb{R}^m$ (orthogonal basis)

(2) Row space: $\operatorname{Ran} A^T = \operatorname{range} V_r = \operatorname{Span}\{v_1, \dots, v_r\} \subset \mathbb{R}^n$

(3) Null-space: $\operatorname{Ker} A = (\operatorname{range} V_r)^\perp = \operatorname{Span}\{v_{r+1}, \dots, v_n\} \subset \mathbb{R}^n$

(4) Left null-space: $\operatorname{Ker} A^T = (\operatorname{range} U_r)^\perp = \operatorname{Span}\{u_{r+1}, \dots, u_m\} \subset \mathbb{R}^m$



↳ SVD reduces A to a diagonal transformation between the row and column spaces equipped with the right orthonormal basis.

Proof ① . $\text{Ran } A \subset \text{Ran } U_r$ ✓

• $u_i = \frac{1}{\sigma_i} A v_i$, $i = 1 \dots r \Rightarrow \text{Span}\{u_1, \dots, u_r\} \subset \text{Ran } A$ etc.

② Recall that $\text{Ker } A = (\text{Ran } A^T)^\perp$

Application Pseudo-inverse !

Since columns of U_r, V_r form orthonormal basis of the column and row space of A respectively:

$$A^+ = V_r \Sigma_r^{-1} U_r^T \qquad \Sigma_r^{-1} = \begin{bmatrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_r^{-1} \end{bmatrix}$$

Recall properties: • $x = A^+ b$ minimizes $\|Ax - b\|_2^2$.

Indeed $AA^+ b = (U_r \Sigma_r V_r^T)(V_r \Sigma_r^{-1} U_r^T) b$

$= \underbrace{U_r U_r^T}_{\perp \text{ projector on Ran } A} b$

- $x = A^+ b$ is $\perp$ to $\mathrm{Ker}\, A$ (i.e. it belongs to the row space of A).
- $AA^+A = A$
- If $\begin{cases} \text{rank } A = n, \\ \text{rank } A = m, \\ \text{rank } A = m = n, \end{cases}$ $\begin{cases} A^+A = I_n \\ AA^+ = I_m, \\ A^+ = A^{-1} \end{cases}$ $\begin{cases} A^+ = (A^TA)^{-1}A^T \\ A^+ = A^T(AA^T)^{-1} \end{cases}$

II) Matrix approximation by the SVD.

Connexion between matrix norms and SVD:

Suppose $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$, $p = \min\{m,n\}$

① Frobenius norm: Recall $\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2}$

Then, $\boxed{\|A\|_F = \|\Sigma\|_F = \left(\sigma_1^2 + \dots + \sigma_p^2\right)^{1/2}}$

Proof: U, V are orthogonal $\Rightarrow$ conserve distances!

$$\|A\|_F^2 = \sum_{j=1}^{n} \|a_j\|_2^2 = \sum_{j=1}^{n} \|U^T a_j\|_2^2 = \|U^TA\|_F^2$$

(a_j: columns of A)

$$= \|\Sigma V^T\|_F^2 = \left\| \begin{bmatrix} \sigma_1 v_1^T \\ \vdots \\ \sigma_p v_p^T \end{bmatrix} \right\|_F^2 = \sigma_1^2 + \dots + \sigma_p^2$$

② Induced 2-norm / Spectral norm

Recall $\|A\|_2 = \max_{\|x\|_2=1} \frac{\|Ax\|_2}{\|x\|_2}$

Now, $\|Ax\|_2^2 = (Ax)^T(Ax)$
$= (U\Sigma V^T x)^T (U\Sigma V^T x)$
$= (\Sigma V^T x)^T U^T U (\Sigma V^T x)$
$= (\Sigma V^T x)^T (\Sigma V^T x)$
$= \|\Sigma y\|_2^2$ where $y = V^T x$

Note $\|\Sigma y\|_2^2 = (\sigma_1 y_1)^2 + \cdots + (\sigma_p y_p)^2$
$\leq \sigma_1^2 (y_1^2 + \cdots + y_p^2) \leq \sigma_1^2 \|x\|_2^2$

Hence $\frac{\|Ax\|_2}{\|x\|_2} \leq \sigma_1$. Furthermore $\frac{\|Av_1\|_2}{\|v_1\|_2} = \sigma_1$

so $\boxed{\|A\|_2 = \sigma_1}$

③ Low-rank approximation.

Given a matrix $A \in \mathbb{R}^{m \times n}$, we have the dyadic form for rank $A = r$:

$$A = \sigma_1 u_1 v_1^T + \cdots + \sigma_r u_r v_r^T$$

Since $\sigma_1 \geq \cdots \geq \sigma_r > 0$, we might hope that partial sum

$$\sum_{j=1}^{k} \sigma_j u_j v_j^T \approx A$$

as missing pieces are small, hopefully for $k \ll r$.

⇝ Especially true if $\sigma_{k+1}, \dots, \sigma_r$ are really small.

Note: Similar idea could be used with eigenvectors for square diagonalizable A:

$$A = V \Lambda V^{-1} = \sum_{j=1}^{n} \lambda_j v_j \hat{v}_j^T$$

v_j columns of V, $\hat{v}_j^T$ rows of V^{-1}

However, no obvious ordering eigenvalues, and eigenvectors not generally orthogonal.
Also, not applicable to non-square matrix.

Example $A = \begin{bmatrix} 2 & 100 \\ 0 & 1 \end{bmatrix}$

- **Eigendecomposition:**

$$A = \begin{bmatrix} 1 & 1 \\ 0 & -1/100 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 100 \\ 0 & -100 \end{bmatrix}$$

$$= 2 \begin{bmatrix} 1 & 100 \\ 0 & 0 \end{bmatrix} + 1 \begin{bmatrix} 0 & -100 \\ 0 & 1 \end{bmatrix}$$

Neither good approximation!

- **SVD:** Computer.

March 22

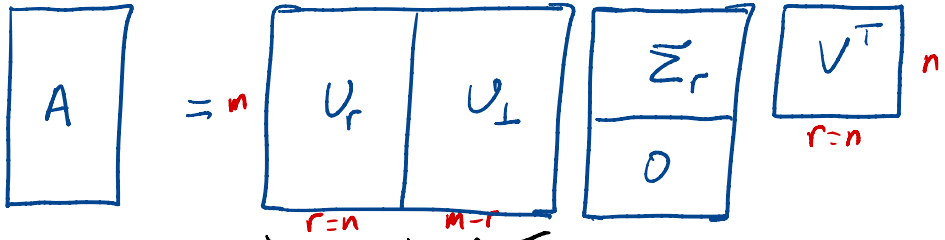
Applications of the SVD

Recall the SVD factorization of $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^T$$

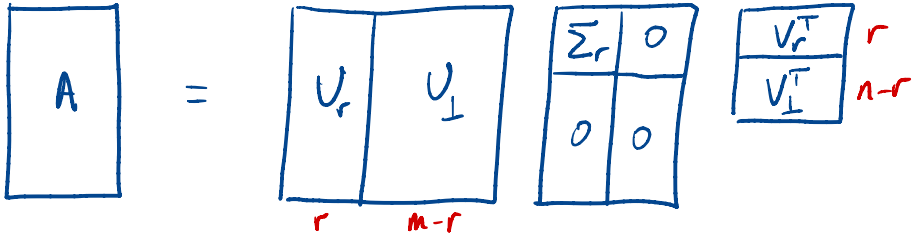
* Full SVD: $\text{rank}(A) = r$ $\quad \Sigma_r = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$

→ Full rank case:

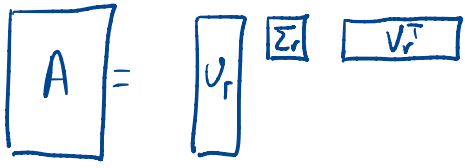


Basis for Range(A) Basis for Ker(A)

→ Rank-deficient: $r < n$



* Reduced SVD:



II Low-rank approximation.

Definition der $A = \sum_{j=1}^{r} \sigma_j u_j v_j^T$ rank-r, in terms of its dyadic decomposition.

We define for $k \leq r$ the truncated singular value decomposition of rank k:

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T = \sigma_1 u_1 v_1^T + \dots + \sigma_k u_k v_k^T$$

Theorem (Schmidt, Mirsky, Eckart-Young)

For $k \leq \operatorname{rank} A$, the truncated SVD A_k is the best rank-k approximation of A:

$$\sigma_{k+1} = \|A - A_k\|_2 = \min_{\operatorname{rank}(X) \leq k} \|A - X\|_2$$

$$(\sigma_{k+1}^2 + \dots + \sigma_p^2)^{1/2} = \|A - A_k\|_F = \min_{\operatorname{rank}(X) \leq k} \|A - X\|_F.$$

Note: $A \approx U_k \Sigma_k V_k^T$

$$U_k = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix} \quad \Sigma_k = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \quad V_k = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$$

$m \times k$ entries; k entries; $n \times k$ entries

Total $(m+n+1)k$ entries. If $k \ll \min(m,n)$ effective compression!

Three types of low-rank matrices:

1) Truly low rank: extreme case uv^T

2) Exponentially decreasing eigenvalues:
"low effective rank"

3) Incomplete matrices, completed to low rank.

→ Examples with Julia.

II) Principal Component Analysis. (PCA)

Fundamental tool of multivariate statistics.

Linear algebraists will say - "PCA is the SVD!"

① A few notions.

- Random variable $X \in \mathbb{R}^n$
- Expected value $E[X]$
  - ↳ Linear function: $E[\alpha X + \beta] = \alpha E[X] + \beta$

- **Variance**: measures expected deviation from mean

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$
$$= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \geq 0$$

- **Covariance**: $\mathrm{Cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right]$
$$= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

Note. • $\mathrm{Cov}(X, X) = \mathrm{Var}(X)$
• $\mathrm{Cov}(X, Y) = 0$ if X, Y independent.

↳ Notions at the heart of PCA: large covariance means the variables are highly correlated.

Now suppose $X_1, \ldots, X_n$ are real-valued random variables in which we suspect redundancy

(ex: stock market).

↳ A few aggregate random variables may capture much of the variance / randomness:

$$Y = \sum_{j=1}^{n} \gamma_j X_j$$

We compute the variance of such a variable:

$$\mathrm{Var}\left(\sum_{j=1}^{n} \gamma_j X_j\right) = E\left[\left(\sum_{j=1}^{n} \gamma_j X_j\right)^2\right] - E\left[\sum_{j=1}^{n} \gamma_j X_j\right]^2$$

$$= E\left[\left(\sum_{i=1}^{n} \gamma_i X_i\right)\left(\sum_{j=1}^{n} \gamma_j X_j\right)\right] - \left(\sum_{i=1}^{n} \gamma_i E[X_i]\right)\left(\sum_{j=1}^{n} \gamma_j E[X_j]\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \gamma_i \gamma_j \left(E[X_i X_j] - E[X_i]E[X_j]\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \gamma_i \gamma_j \, \mathrm{Cov}(X_i, X_j).$$

Let's Define covariance matrix C, $c_{ij} = \mathrm{Cov}(X_i, X_j)$

and $v = [\gamma_1, \dots, \gamma_n]^T$ then

$$\mathrm{Var}(Y) = v^T C v \geq 0.$$

Hence C is symmetric and semi-definite positive.

Another expression: $C = E[X X^T] - E[X]E[X]^T$.

(2) Principal components.

New problem: maximize variance $v^T C v$ with constraint $\gamma_1^2 + \dots + \gamma_n^2 = 1$.

↳ solution: $v^T C v = \|C\|_2 = \lambda_1$

for v_1 unit singular / eigen-vector associated with
λ_1 the largest singular / eigen-value of C.

$$\lambda_1 = v_1^T C v_1 = \max_{\|v\|=1} v^T C v.$$

Resulting **leading principal component**

$$v_1^T X = \gamma_1 X_1 + \dots + \gamma_n X_n$$

has maximum variance.

Next, we would like another combination $w^T X$, uncorrelated with the first:

$$\mathrm{Cov}(w^T X, v^T X) = 0$$

with maximum variance again.

↳ Compute

$$\mathrm{Cov}(w^T X, v_1^T X) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i v_j \, \mathrm{Cov}(X_i, X_j)$$

$$= w^T C v_1 = \sigma_1 (w^T v_1)$$

↳ **uncorrelated means $w^T v_1$ orthogonal.**

From SVD, this means

$$\max_{w \perp v_1} w^T C w = \lambda_2 \quad \text{for} \quad C v_2 = \lambda_2 v_2$$

i.e. second singular pair.

Following this pattern:

Let $X = [X_1, \dots, X_n]$ collect n random variables, with covariance matrix $C \in \mathbb{R}^{n \times n}$.
Since C is symmetric positive semi-definite, it has real, positive eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$ and orthonormal eigenvectors $v_1, \dots, v_n$.

The **k-th principal component** is the aggregate variable

$$Y_k = v_k^T X$$

Variables $Y_1, \dots, Y_n$ are uncorrelated and have variance

$$\mathrm{Var}(Y_k) = v_k^T C v_k = \lambda_k .$$

↳ If later eigenvalues are small, most of the randomness is "explained" by a smaller set of principal components.

(Another view on "low rank").

⇝ How many principal components are needed to describe data set? Since $Y_1, \dots, Y_n$ uncorrelated,

$$\mathrm{Var}(Y_1 + \dots + Y_k) = \mathrm{Var}(Y_1) + \dots + \mathrm{Var}(Y_k) = \lambda_1 + \dots + \lambda_k$$

↳ Proportion of variance captured is

$$\frac{\left(\sum_{j=1}^{k} \lambda_k\right)}{\left(\sum_{j=1}^{n} \lambda_n\right)}$$

↝ Pick k large enough that ratio is e.g. above 0.75 to explain more than 75% of variance.

③ Approximate PCA from data.

Idea: in practice, need to estimate mean and covariance from data.

Suppose we have m samples of $X_1, \ldots, X_n$:

$$x_{j,k} \quad j = 1, \ldots, n \quad k = 1 \ldots m$$

the k-th sample of the j-th variable.

① Estimate of the expected value:

$$\mu_j = \frac{1}{m} \sum_{k=1}^{m} x_{j,k}$$

It is an unbiased estimate (i.e. $E[\mu_j] = E[X_j]$)

② Estimate of covariance:

$$\mathrm{Cov}(X_i, X_j) = E\left[(X_i - E[X_i])(X_j - E[X_j])\right]$$

Natural (but wrong) guess would be

$$1/m \sum_{k=1}^{m} (x_{i,k} - \mu_i)(x_{j,k} - \mu_j)$$

but it is biased, i.e. expectation value $\neq \mathrm{Cov}(X_i, X_j)$.

Better is the unbiased estimate

$$s_{j,k} = \frac{1}{m-1} \sum_{k=1}^{m} (x_{i,k} - \mu_i)(x_{j,k} - \mu_j)$$

This is an inner product $s_{j,k} = \frac{1}{m-1} (x_i - \mu_i)^T (x_j - \mu_j)$

Let

$$\mathbb{X} = \begin{bmatrix} (x_1 - \mu_1) & \cdots & (x_n - \mu_n) \end{bmatrix}$$

i.e. we center samples of each variable about their empirical mean, then

Empirical covariance matrix: $S = [s_{j,k}] = \frac{1}{m-1} \mathbb{X}^T \mathbb{X}$

Lo Eigenvectors of S lead to sample principal components.

Link to SVD: Link $S \leftrightarrow X^T X$

as if $X = U \Sigma V^T$ (SVD of data matrix)

then $S = \frac{1}{m-1} (U \Sigma V^T)^T (U \Sigma V^T)$

$= V \left(\Sigma^T \Sigma / m-1 \right) V^T$

PCA is just the SVD of (centered) data matrix !.

Step 1 Collect m samples of each random variable.

$X = [x_{i,j}]_{m \times n}$ Usually $m >> n$

Step 2 Empirical means of the columns

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{i,j}$$

Step 3 Form Mean-center data matrix

$$\underline{X} = \begin{bmatrix} x_{1,1} - \mu_1 & \cdots & x_{1,n} - \mu_n \\ & & \\ x_{m,1} - \mu_1 & \cdots & x_{m,n} - \mu_n \end{bmatrix}$$

Step 4 Compute reduced SVD $\underline{X} = U \Sigma V^T$,

$U \in \mathbb{R}^{m \times n}$, $\Sigma = \text{Diag}(\sigma_1, \dots, \sigma_n)$, $V \in \mathbb{R}^{n \times n}$

Step 5 Form sample principal components $Y_k = v_k^T X$

where v_k k-th column of V.

Step 6 Assess importance of principal components via eigenvalues of S,

$$\lambda_k = \sigma_k^2/(m-1).$$

If rapid decay → need only first few PCs.

⚠ Scale of columns matters. Changing units may alter principal components greatly.

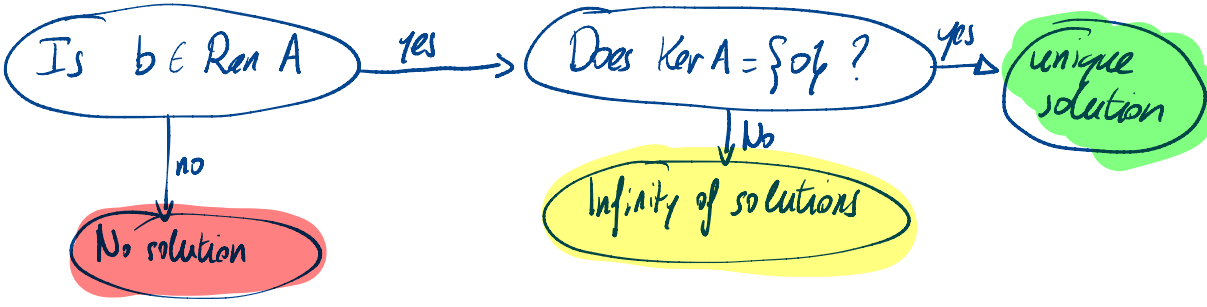
④ Clustering via PCA.

Example: wine data set.

Tuesday, March 29

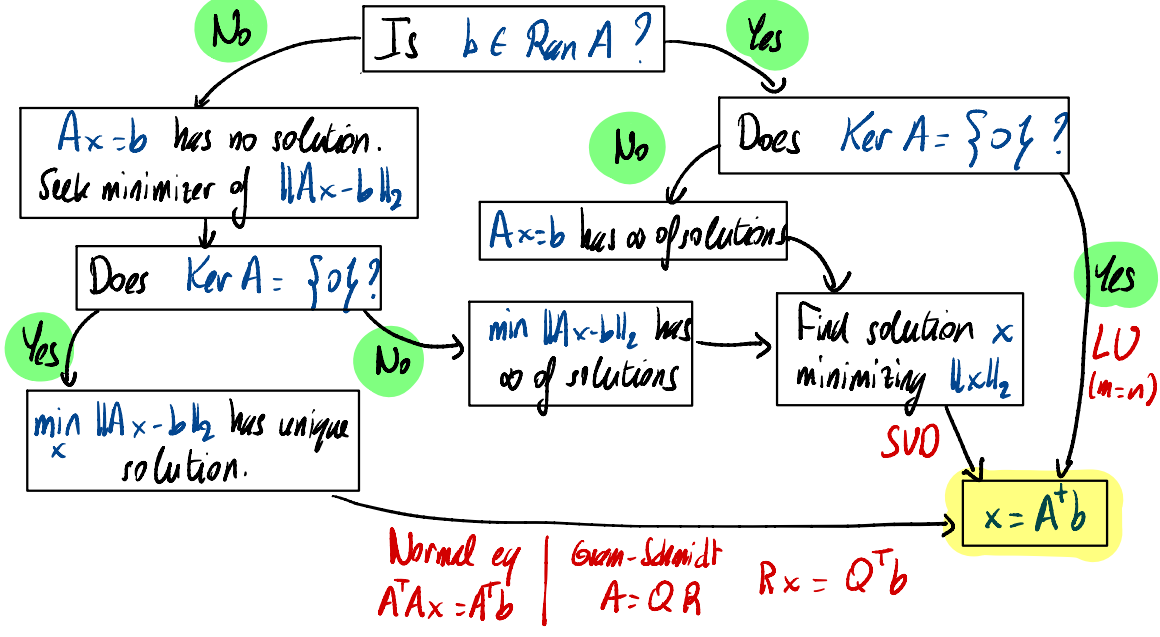
Return to least squares.

I) Recall basic flowchart about linear problem: $Ax = b$



When no solution exists, we can instead turn to the least squares problem:

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



Example $A = \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

↳ Here rank $A = 2 = n$

① $b \notin \text{Ran } A$

② . Normal equations:

$$A^TA = \begin{bmatrix} 2 & 1 \\ 1 & 5 \end{bmatrix} \qquad A^Tb = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$\begin{bmatrix} \boxed{2} & 1 \\ 1 & 5 \end{bmatrix}$ $L = \begin{bmatrix} 1 & 0 \\ 1/2 & 1 \end{bmatrix}$ $A^{(1)} = U = \begin{bmatrix} 2 & 1 \\ 0 & 9/2 \end{bmatrix}$

↑ multiplier

↳ solve $Ly = b$:

$$\begin{aligned} y_1 &= 2 \\ y_1/2 + y_2 &= 3 \end{aligned} \quad \Big\} \quad \begin{aligned} y_1 &= 2 \\ y_2 &= 2 \end{aligned}$$

↳ solve $Ux = y$:

$$\begin{aligned} 2x_1 + x_2 &= 2 \\ 9/2\, x_2 &= 2 \end{aligned} \quad \Big\} \quad \begin{aligned} x_1 &= 7/9 \\ x_2 &= 4/9 \end{aligned}$$

• QR: $A = \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}$ $a_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ $a_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

1) $\|a_1\| = \sqrt{2}$ $q_1 = \frac{1}{\|a_1\|} a_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

2) $v_2 = a_2 - (q_1, a_2) a_1$

$= \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$

$1/\sqrt{2}$

$= \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 2 \\ -1/2 \end{bmatrix}$

$\|v_2\| = \sqrt{1/4 + 4 + 1/4} = \sqrt{9/2} = 3/\sqrt{2}$

$q_2 = \begin{bmatrix} 1/3\sqrt{2} \\ 2\sqrt{2}/3 \\ -1/3\sqrt{2} \end{bmatrix}$

Note $a_1 = \sqrt{2}\, q_1$ and $a_2 = \frac{1}{\sqrt{2}} q_1 + \frac{\sqrt{3}}{2} q_2$

$\Rightarrow$ $A = \begin{bmatrix} 1/\sqrt{2} & 1/3\sqrt{2} \\ 0 & 2\sqrt{2}/3 \\ 1/\sqrt{2} & -1/3\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & 3/\sqrt{2} \end{bmatrix}$

Q R

Now $Q^T b = \begin{bmatrix} 2\sqrt{2} \\ 2\sqrt{2}/3 \end{bmatrix} = y$

Solve $Rx = y$:

$$\begin{cases} \sqrt{2}\, x_1 + 1/\sqrt{2}\, x_2 = 2\sqrt{2} \\ 3/\sqrt{2}\, x_2 = 2\sqrt{2}/3 \end{cases}$$

$$\Rightarrow \begin{cases} x_2 = 4/9 \\ x_1 = 1 - x_2/2 = 7/9 \end{cases} \qquad \text{so} \quad x = \begin{bmatrix} 7/9 \\ 4/9 \end{bmatrix}$$

③ Pseudo inverse: here $A^+ = (A^TA)^{-1}A^T$

We have computed $A^TA = \begin{bmatrix} 2 & 1 \\ 1 & 5 \end{bmatrix}$

$$\hookrightarrow (A^TA)^{-1} = \frac{1}{9}\begin{bmatrix} 5 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 5/9 & -1/9 \\ -1/9 & 2/9 \end{bmatrix}$$

$$A^+ = \begin{bmatrix} 5/9 & -1/9 \\ -1/9 & 2/9 \end{bmatrix}\begin{bmatrix} 1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4/9 & -2/9 & 5/9 \\ 1/9 & 4/9 & -1/9 \end{bmatrix}$$

II) Stability of the pseudo-inverse.

Preliminary: the SVD itself is largely stable i.e.

$$A = USV^T \qquad A + \varepsilon P = U_\varepsilon S_\varepsilon V_\varepsilon^T$$

$$\hookrightarrow \quad \| S - S_\varepsilon \| \sim \varepsilon.$$

Example: take $A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ (rank 1)

$$\rightarrow \quad A = \underbrace{\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sqrt{10} \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{V^T}$$

Now we take a look at the pseudo-inverse:

$$A^+ = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{10} \end{bmatrix} \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}$$

$$= \begin{bmatrix} 1/10 & 1/5 \\ 1/10 & 1/5 \end{bmatrix}$$

Small tweak: $A_\varepsilon = \begin{bmatrix} 1 & 1 \\ 2 & 2+\varepsilon \end{bmatrix}$

Now $\operatorname{rank} A_\varepsilon = 2$ for $\varepsilon > 0$: it is invertible

$$A_\varepsilon^+ = A_\varepsilon^{-1} = \frac{1}{\varepsilon} \begin{bmatrix} 2+\varepsilon & -1 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} 2/\varepsilon + 1 & -1/\varepsilon \\ -2/\varepsilon & 1/\varepsilon \end{bmatrix}$$

Note that as $\varepsilon \to 0$, A_ε^+ gains large entries.

In fact $A_\varepsilon^+ \not\to A^+$ as $\varepsilon \to 0$

↳ Not continuous function of matrix entries.

Problem is that $A^+ = \sum_{j=1}^{r} \frac{1}{\sigma_j} v_j u_j^T$

↳ if $\sigma_j(\varepsilon) \to 0$ then corresponding term blows up.

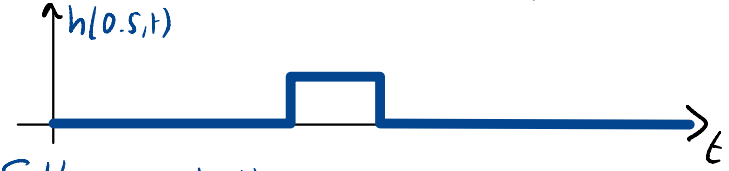
Example: barcode experiment. Deblurring

↝ convolution with blurring kernel:

$$\int_0^1 h(s,t) f(t)\, dt = b(s)$$

f: exact; b: blurry observation

Examples:



$$h(s,t) = \begin{cases} 1/2z & |s-t| < z \\ 0 & |s-t| > z \end{cases}$$

↝ transforms a point source into a spot of width z.

Compute $b(s) = \int_0^1 h(s,t) f(t)\, dt = \int_{s-z}^{s+z} \frac{f(t)}{2z}\, dt$

$= \frac{\text{integral of } f \text{ in } [s-z, s+z]}{\text{length of interval } [s-z, s+z]}$

$=$ average of f in $[s-z, s+z]$.

↳ observation is a moving average of A.

Problem: blurring removes small details, and maps two different f_1, f_2 to similar observations b_1, b_2.
↳ inverse problem is difficult to solve (ill-posed.)

Discretization: record usually set of n pixels.

Discrete points: $s_j = \frac{j-1/2}{n}$ $\quad t_k = \frac{k-1/2}{n}$ $\quad j,k = 1 \dots n$.

Approximate integral by midpoint quadrature rule:

$$\frac{1}{n} \sum_{k=1}^{n} h(s_j, t_k) f_k = b_j \quad \text{where} \quad \begin{cases} f_k = f(t_k) \\ b_j = b(t_j) \end{cases}$$

↳

$$\underbrace{\frac{1}{n} \begin{bmatrix} h(s_1,t_1) & \cdots & h(s_n,t_1) \\ \vdots & & \vdots \\ h(s_n,t_1) & & h(s_n,t_n) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}}_{b}$$

Problem: A invertible, but close to rank-deficient.

In the presence of noise, $A^{-1} b_{noise}$ produces garbage.

Analysis: $f_{noise} = A^{-1} b_{noise} = \sum_{j=1}^{n} \frac{(u_j^T b_{noise})}{\sigma_j} v_j$

→ in general, $u_j^T b$ decreases for small σ_j but not $u_j^T b_{noise}$.

Example: with barcode. Show some singular vectors for large j → highly oscillatory.

III Regularization for ill-posed problems.

~> Truncation: $x = A_k^+ b$, $A_k^+ = \sum_{j=1}^{k} 1/\sigma_k \, v_j u_j^T$

~> Tikhonov problem: $\min_x \|Ax - b\|_2^2 + \delta^2 \|x\|_2^2$.

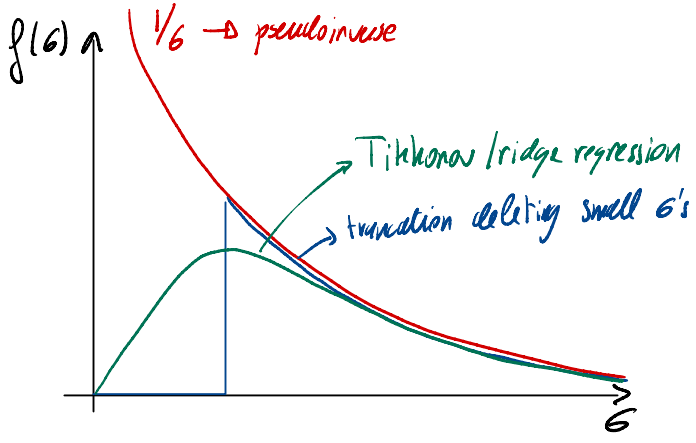
$x = A_\delta^+ b$, $A_\delta^+ = (A^T A + \delta^2 I)^{-1} A^T$

"Ridge regression"

$A_\delta^+ = \sum_{j=1}^{n} \frac{\sigma_j}{\sigma_j^2 + \delta^2} v_j u_j^T$

Compare with pseudo-inverse: $A^+ = \sum_{j=1}^{r} \frac{1}{\sigma_j} v_j u_j^T$

Each approach differs by a different filter function applied to the singular values!



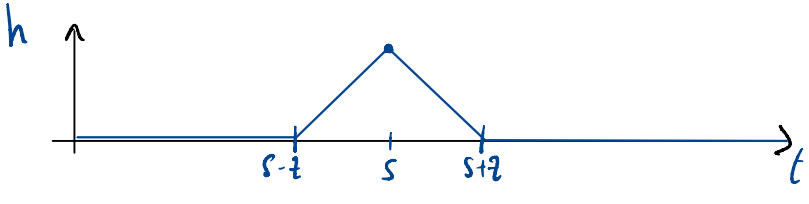
Note as $\delta \to 0$, Tikhonov filter $\frac{\sigma}{\sigma^2+\delta^2}$ converges to $1/\sigma$

How to pick this regularization parameter ?

↳ L-curve.

Example . Hat-function kernel blurring

$$h(s,t) = \frac{1}{z}\max\left(0,\ 1-\frac{|s-t|}{z}\right)$$

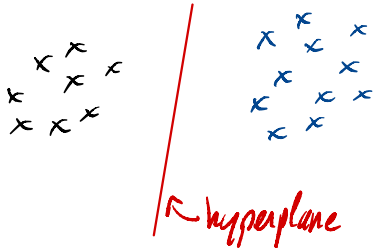


Tuesday, April 12

OPTIMIZATION: INTRO.

Motivation. Support Vector Machines (SVM)

classification



↖ hyperplane $w^T x + b = 0$

↳ find $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ such that

$w^T x + b < 0$ (resp. > 0)

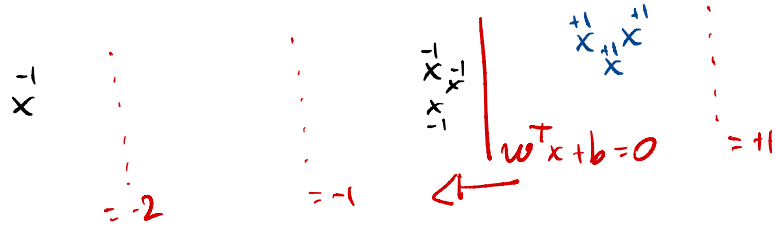
for x ∈ group 1 (resp. group 2).

First idea: least squares. Given data $X \in \mathbb{R}^{m \times n}$ (rows $x_1, \dots, x_m$) with labels $y = [y_1, \dots, y_m] \in \{-1, 1\}^m$

we find w, b such that

$$\begin{bmatrix} x_1^T w + b \\ \vdots \\ x_m^T w + b_m \end{bmatrix} = \begin{bmatrix} X & | & 1 \end{bmatrix} \begin{bmatrix} w \\ \hline b \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

Problem: "outliers" are not ideal. See:

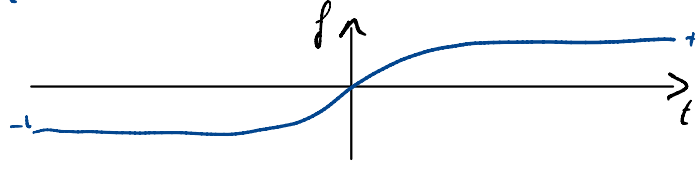


↳ least-squares cost "punishes" large values of model $|w^T x + b| \gg 1$ on data.

↳ better idea: use non-linear function

$f(w^T x + b)$

with f:



$\Rightarrow$ minimize cost function $F(w,b) := \sum_{i=1}^{m} |f(w^T x_i + b) - y_i|^2$

↳ Problem: not a linear LS problem anymore.
How to find minimum?

I) Optimization basics.

Goal: minimize / maximize function $F(x_1, \dots, x_n)$ with many variables.

Possibly constraints: $\vec{x} \in K \subset \mathbb{R}^n$.

Expression "argmin" Minimum of $F(x) = (x-1)^2$ is zero.

Often, we are more interested in where is the minimum: argument x^* such that $F(x^*) = \min F$.

→ Not always unique.

1) **Calculus:** Derivatives $= 0$ at minimum x^* (assuming no constraints).

↳ partial derivatives: $\frac{\partial F}{\partial x_i} = 0, \quad i = 1 \dots n.$

Gradient $\nabla F = \left[\frac{\partial F}{\partial x_i}\right]_{\partial x_i} = 0.$

Key calculus fact: Approximation by Taylor expansions.

(a) One function, one variable.

$$F(x + \delta x) \approx F(x) + \delta x \frac{dF}{dx}(x) + \frac{\delta x^2}{2} \frac{d^2F}{dx^2}(x)$$

(b) One function F, many variables $x_1, \dots, x_n$

$$F(\vec{x} + \vec{\Delta x}) \approx F(\vec{x}) + \Delta x^T \nabla F(x) + \frac{1}{2} \Delta x^T H(x) \Delta x$$

where ∇F: gradient $\left[\frac{\partial F}{\partial x_i}\right]$ → vector

H: Hessian $\left[\frac{\partial^2 F}{\partial x_i \partial x_j}\right]$ → symmetric matrix

Graph of F is a surface in dimension $n+1$.

(c) m functions $f_1, \dots, f_m$ (vector function)
n variables $x_1, \dots x_n$

then $\vec{f}(\vec{x} + \vec{\Delta x}) \approx \vec{f}(\vec{x}) + J(\vec{x}) \vec{\Delta x}$

with J: Jacobian matrix whose rows contain the gradients of $f_1, \dots, f_m$

$$J = \begin{bmatrix} (\nabla f_1)^T \\ \vdots \\ (\nabla f_m)^T \end{bmatrix} = \left[\frac{\partial f_i}{\partial x_j}\right]_{i,j}$$

Note: H Hessian of F ↔ Jacobian of ∇F.

(det J useful for n-dimensional change of variables!)

(2) Minimum, Convexity.

Minimization problems often feature:

- cost $F(x_1, \dots, x_n)$ with many variables
- Constraints on x:
  - * Linear constraints $Ax = b$
  - * Inequality constraints $x \geq 0$
  - * Integer/Binary constraints Each x_i is 0 or 1

Note: linearity is not compatible with minimization.

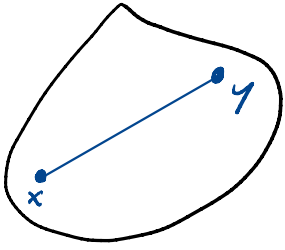
Similarly crucial role played by convexity.

Definitions.

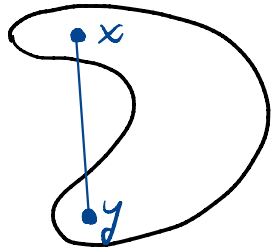
- K is a convex set: if $x, y \in K$, so is the line from x to y.
- F is a convex function: set of points on or above graph of F is convex
- F is smooth and convex: $F(x) \geq F(y) + \nabla F(y) \cdot (x - y)$

  tangent at y

Sets

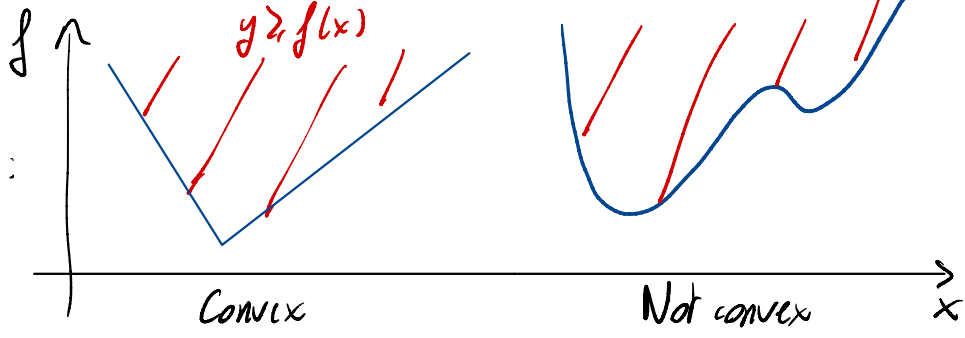


CONVEX



NOT CONVEX

Functions:

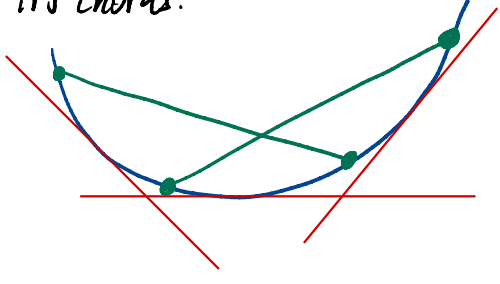


Alternative characterization:

- F is **convex** if $F(tx + (1-t)y) \leq t\,F(x) + (1-t)\,F(y)$ for $0 \leq t \leq 1$.

Strictly convex if true for all strict inequalities.

- F is convex if it stays above its tangents, but below its chords.



Property

- The intersection of a family of convex sets is a convex set.
- The maximum of a (finite) set of convex functions is convex.

Not true for unions / minimum of convex functions!

Example. The set of positive definite, $n \times n$ matrices is convex.

Proof. $x^T(tA + (1-t)B)x$

$= t\,x^TAx + (1-t)\,x^TBx \geq 0.$

Case of smooth functions

Recall: if F is twice differentiable,

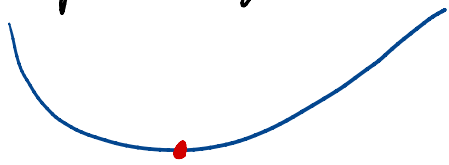
$$F(y) \approx F(x) + (y-x)^T \nabla F(x) + \frac{1}{2}(y-x)^T H(x)(y-x)$$

$\Rightarrow$ Convex if $H(x) \geq 0$

Convexity test. $F(x_1, \dots, x_n)$ smooth is convex if and only if its Hessian matrix $H(x)$ is positive, semi-definite $\forall x$.

F is strictly convex if $H(x)$ is positive definite $\forall x$.

③ Minimization of convex functions.



→ There cannot be two isolated minima:

if $F(x) = F(y) = m = \min F$

then for $t \in (0,1)$, $z = tx + (1-t)y$,

$m \leq F(z) \leq tF(x) + (1-t)F(y) = m.$

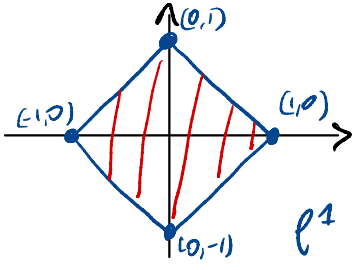
In fact, **the set of minimizing points is convex.**

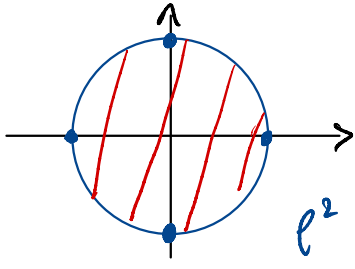
Examples. ℓ^1, ℓ^2, ℓ^∞ balls.

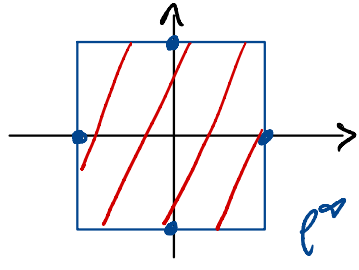
Vector norms are by construction convex functions:

$$\|tx + (1-t)y\| \leq t\|x\| + (1-t)\|y\|.$$

Consequently, unit balls are also convex:







Newton's method. For convex functions,

$x^* \in \operatorname{argmin} F$ iff $\nabla F(x^*) = 0$.

Suppose we have an approximation $x_k \approx x^*$.

How to improve it? Near x_k,

$$\nabla F(x) \approx \nabla F(x_k) + J(x_k)(x_{k+1} - x_k).$$

↳ make RHS zero:

$$\begin{cases} H(x_k)\,\Delta x_k = -\nabla F(x_k) \\ x_{k+1} = x_k + \Delta x_k \end{cases}$$

Other POV: x_{k+1} minimizes quadratic approximation

$$F(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

This is Newton's method.

Close to x^* it converges quadratically:

$$\|x_{k+1} - x^*\| \le C \|x_k - x^*\|^2.$$

Further away, not so good – allow backtracking:

$x_{k+1} = x_k + t\,\Delta x_k$ ↳ reduce $t \le 1$ until

$F(x_k + t\,\Delta x_k) \le F(x_k) + \alpha t \nabla F(x_k)^T \Delta x_k$.

α: parameter, $\alpha < 1/2$

Conclusion: • Newton is fast. Uses 2nd derivatives.

• Newton is (often) too expensive for same reason.

Cheaper alternative: gradient descent.

Nonlinear least-squares.

Idea: if $F(x) = \|Ax-b\|^2$ fast solution method is known.

What if we have m nonlinear equations to solve:

$$\hat{y}(p) \approx y$$

where $\hat{y}(p) \in \mathbb{R}^m$ depends nonlinearly on $p \in \mathbb{R}^n$.

$$\Rightarrow E(p) = \sum_{i=1}^{m} (y_i - \hat{y}_i(p))^2$$

$$= y^T y - 2y^T \hat{y}(p) + \hat{y}(p)^T \hat{y}(p)$$

(Possible to include weighting matrix W).

To minimize $E(p)$, we compute

$$\nabla E = 2J(p)^T (y - \hat{y}(p))$$

with the Jacobian matrix $J = \frac{dy}{dp} \in \mathbb{R}^{m \times n}$.

We seek to solve $\nabla E = 0$.

↳ Approximate Newton: Normal equations

$$J^T J (p_{n+1} - p_n) = J^T (y - \hat{y}(p_n))$$

(note $2J^TJ \approx$ Hessian (E) :

$$E(p+\Delta p) \approx (y - \hat{y}(p) - J\Delta p)^T (y - \hat{y}(p) - J\Delta p)$$

$$\approx \quad \dots \quad + \quad \Delta p^T J^T J \Delta p.$$

↳ Stabilization: Tikhonov regularization

$$(J^TJ + \lambda I)(p_{n+1} - p_n) = J^T (y - \hat{y}(p_n))$$

Levenberg - Marquardt

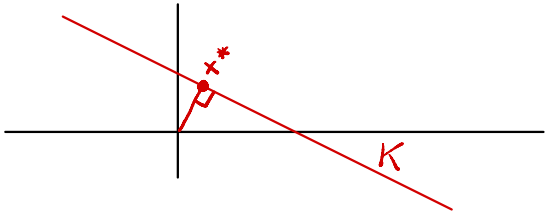
- Usually start with large λ ($\approx$ gradient descent) then reduce it.
- Not exactly Newton (Gauss-Newton iteration for $\lambda = 0$)

2) Constraints and Lagrange multipliers.

Example Minimize $F(x) = x_1^2 + x_2^2$

on line $K := \{ a_1 x_1 + a_2 x_2 = b \}$

$g(x) = a_1 x_1 + a_2 x_2 - b = 0$



To bring the constraint into the cost function:

Define the Lagrangian

$$L(x, \lambda) := F(x) + \lambda\, g(x)$$

(λ: unknown multiplier)

$$= x_1^2 + x_2^2 + \lambda (a_1 x_1 + a_2 x_2 - b)$$

Set all 3 derivatives $\frac{\partial L}{\partial x_1}$, $\frac{\partial L}{\partial x_2}$ and $\frac{\partial L}{\partial \lambda} = 0$.

Solve all 3 equations for x_1, x_2, λ.

$$\begin{cases} 2x_1 + a_1 \lambda = 0 \\ 2x_2 + a_2 \lambda = 0 \\ a_1 x_1 + a_2 x_2 - b = 0 \end{cases} \rightarrow \text{Constraint!}$$

Now $x_1 = -\frac{1}{2} \lambda a_1$ and $x_2 = -\frac{1}{2} \lambda a_2$

$$\Rightarrow \quad -\frac{1}{2} \lambda (a_1^2 + a_2^2) = b \quad \text{and} \quad \lambda = \frac{-2b}{a_1^2 + a_2^2}.$$

This yields $\boxed{x_1^* = \frac{a_1 b}{a_1^2 + a_2^2}}$ and $\boxed{x_2^* = \frac{a_2 b}{a_1^2 + a_2^2}}$

with $F(x^*) = (x_1^*)^2 + (x_2^*)^2 = \frac{b^2}{a_1^2 + a_2^2}$.

In particular, $\frac{d}{db}\left(F(x^*)\right) = \frac{2b}{a_1^2 + a_2^2} = -\lambda$.

The value of the Lagrange multiplier w.r.t. constraint level b is $-\lambda$.

Why this works? At minimum, $\nabla F(x^*) \perp K$, $\nabla g(x^*) \perp K$

(unique since F, K convex)

$$\begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \frac{\partial L}{\partial x_2} \end{bmatrix} = \nabla f + \lambda \nabla g$$

$\rightsquigarrow$ there is λ such that $= 0$.
iff ∇f, ∇g proportional

i.e. constraint line tangent to circle.

Minimizing quadratic function with a linear constraint.

Look for $x \in \mathbb{R}^n$ minimizing $F(x) = \frac{1}{2} x^T S x$

where S is symmetric, positive, definite, subject to

$$A^T x = b \quad \text{with} \quad A \in \mathbb{R}^{n \times m}, \; b \in \mathbb{R}^n$$

with $m < n$, usually.

To form the Lagrangian we introduce m Lagrange multipliers: $\lambda = [\lambda_1, \dots, \lambda_m]^T$ and

$$L(x, \lambda) = \frac{1}{2} x^T S x + \lambda^T (A^T x - b).$$

Note $L(x+\delta x, \lambda + \delta\lambda) = \frac{1}{2}(x+\delta x)^T S (x+\delta x) + (\lambda+\delta\lambda)^T (A^T(x+\delta x) - b)$

$$= \frac{1}{2} x^T S x + \frac{1}{2}\left(\delta x^T S x + x^T S \delta x\right) + \delta x^T S \delta x$$
$$+ \lambda^T (A^T x - b) + \lambda^T A^T \delta x + \delta\lambda^T (A^T x - b) + \delta\lambda^T A^T \delta x$$

$$= L(x, \lambda) + \delta x^T (S x + A\lambda) + \delta\lambda^T (A^T x - b) + \delta x^T S \delta x + \delta\lambda^T A^T \delta x.$$

$\Rightarrow$ $m+n$ derivatives:

$$\begin{cases} x\text{-derivatives}: & \dfrac{\partial L}{\partial x} = S x + A\lambda \\ \lambda\text{-derivatives}: & \dfrac{\partial L}{\partial \lambda} = A^T x - b \end{cases}$$

Now set $\frac{\partial L}{\partial x}(x^*) = 0$, $\frac{\partial L}{\partial \lambda}(x^*) = 0$.

↳ constraint satisfied.

∇F belongs to $\text{Range}(A)$

↳ if $\delta x \in \text{Ker } A^T$ then $\delta x \perp \nabla F$.

Solution x^* minimizes F on K !

Solve: $x^* = -S^{-1}A\lambda^*$

$$\Rightarrow -A^T(S^{-1}A\lambda^*) - b = 0$$

$$\Rightarrow \boxed{\begin{cases} \lambda^* = -(A^TS^{-1}A)^{-1}b \\ x^* = S^{-1}A(A^TS^{-1}A)^{-1}b \end{cases}}$$

Minimum cost:

$$F(x^*) = \frac{1}{2} b^T (A^TS^{-1}A)^{-1}A^TS^{-1}SS^{-1}A(A^TS^{-1}A)^{-1}b$$

$$= \frac{1}{2} b^T (A^TS^{-1}A)^{-1}A^TS^{-1}A(A^TS^{-1}A)^{-1}b$$

$$\boxed{F(x^*) = \frac{1}{2} b^T (A^TS^{-1}A)^{-1}b}$$

Gradient: $\boxed{\frac{\partial F(x^*)}{\partial b} = (A^TS^{-1}A)^{-1}b = -\lambda^*}$

↝ Again, Lagrange multipliers are derivatives of the optimal cost.

KKT matrix. Note

$$\underbrace{\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix}}_{\text{saddle-point matrix.}} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

Block-row elimination yields:

$R_2 - A^T S^{-1} R_1 \rightarrow$

$$\begin{bmatrix} S & A \\ 0 & -A^T S^{-1} A \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

$-A^T S^{-1} A$: Schur complement.

S: positive definite $\Rightarrow$ n positive pivots

$-A^T S^{-1} A$: negative definite $\Rightarrow$ m negative pivots

$\Rightarrow$ (x^*, λ^*) SADDLE POINT of $\mathcal{L}(x, \lambda)$.

Convex in x, concave in λ.

Minimax principle

At fixed λ, minimize for $x^*(\lambda)$:

$$\min_x \frac{1}{2} x^T S x + \lambda^T (A^T x - b)$$

$$\Leftrightarrow \quad S x + A\lambda = 0, \quad \boxed{x^*(\lambda) = -S^{-1} A \lambda}$$

$$\Rightarrow L(x^*(\lambda), \lambda) = \frac{-1}{2} \lambda^T A^T S^{-1} A \lambda - \lambda^T b$$

Then, maximize that minimum:

$$\Leftrightarrow \quad A^T S^{-1} A \lambda^* + b = 0$$

$$\boxed{\lambda^* = -(A^T S^{-1} A)^{-1} b}$$

$$F(x^*) = \max_\lambda \min_\lambda L(x, \lambda) = \frac{1}{2} b^T (A^T S^{-1} A)^{-1} b$$

Similarly we have $F(x^*) = \min_x \max_\lambda L(x, \lambda)$.

$\Leftrightarrow$ At saddle point, $\frac{\partial L}{\partial \lambda} = 0$, $\frac{\partial L}{\partial x} = 0$, and

$$\max_\lambda \min_x L = \min_x \max_\lambda L.$$

Thursday, April 21

Optimization and Gradients.

Reminder Abstract problem:

$$m = \min_x x^T S x \quad \text{subject to} \quad A^T x = b$$

Lagrangian: $L(x, y) = x^T S x + y^T (A^T x - b)$

$$\rightarrow \quad m = \max_y \underbrace{\min_x L(x, y)}_{\text{Primal problem}} = \min_x \underbrace{\max_y L(x, y)}_{\text{Dual problem}}$$

I) Applications

① Example: control of linear ODEs.

$$y'(t) = \underbrace{A(t)}_{\text{system matrix}} y(t) + \underbrace{u(t)}_{\text{control}}$$

e.g., linearization around equilibrium of trajectory (e.g. James Webb observatory around L2)

Goal: using control: least amount of fuel, ensure trajectory

Ex: $\min_u \|u\|_2^2$ subject to $\begin{cases} y(0) = y_0, \\ y(T) = y_f. \end{cases}$

How to find such a u in practice?

Discretization:

$t_0 = 0, \quad t_1 = h, \quad \dots, \quad t_N = T \ (t_n = nh)$

$y_0 \approx y(t_0), \ y_1 \approx y(t_1), \ \dots, \ y_N \approx y(t_N)$

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} A(t)\,y(t) + u(t)\,dt$$

↳
$$y_{n+1} = y_n + \frac{h}{2}\left(A(t_n)\,y_n + A(t_{n+1})\,y_{n+1} + u_n + u_{n+1}\right)$$

Or more complicated / accurate version

↳ linear relation:
$$y_N = \Phi_T\, y_0 + b^T u$$

Lead to a quadratic problem with linear constraint:

$$\min_{u=[u_0,\dots,u_N]^T} \|u\|_2^2 \quad \text{subject to} \quad b^T u = y_N - \Phi_T y_0.$$

Solution using the Lagrangian:

$$\mathcal{L}(u,\lambda) = u^T u + \lambda\left(b^T u - (y_N - \Phi_T y_0)\right)$$

↳ KKT matrix and saddle-point system:

$$\begin{bmatrix} I & b \\ b^T & 0 \end{bmatrix} \begin{bmatrix} u^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 0 \\ y_N - \Phi_T y_0 \end{bmatrix}$$

② Basis pursuit

Given a highly underdetermined problem

$$Ax = b, \quad A \in \mathbb{R}^{m \times n}, \quad m \ll n,$$

find a solution x with many zero components with

Minimize $\|x\|_1 = |x_1| + \dots + |x_n|$ subject to $Ax = b$.

↳ convex, but not quadratic problem. $\|x\|_1$ piecewise linear.

Similar to LASSO:

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 + \lambda \|x\|_1$$

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 \text{ s.t. } \|x\|_1 \leq t.$$

Big picture for algorithms. Consider split problem

Minimize $F_1(x) + F_2(x)$ for x in K

where F_1, F_2, K convex.

① Lagrangian

Consider $\min_x f(x)$ s.t. $Ax = b$

↳ Lagrangian $L(x, y) = f(x) + y^T(Ax-b)$

↝ Saddle point problem $\min_x \max_y L = \max_y \min_x L,$

with $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = 0$

Primal problem: given y, $\min_x L(x,y)$

Dual problem: given x, $\max_y L(x,y)$

Practical algorithm: maximize $m(y) = L(x^*(y), y)$

Look for y^* where $\nabla m = A x^*(y) = 0$ by following gradient uphill: given iterates x_k, y_k,

① Find x_{k+1}: $x_{k+1} = \arg\min_x L(x, y_k)$

② Follow $\nabla m = A x_{k+1} - b$: $y_{k+1} = y_k + s_k (A x_{k+1} - b)$

Key: stepsize s_k carefully chosen.

② Dual decomposition: Suppose $f(x)$ separable into components:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N)$$

(x_1: pieces of x vector)

Write corresponding pieces $A = [A_1 \cdots A_N]$. Then

$$L(x,y) = f(x) + y^T(Ax - b)$$

$$= \sum_{i=1}^{N} \underbrace{\left[f_i(x_i) + y^T A_i x_i - \frac{1}{N} y^T b \right]}_{L_i(x_i, y)}$$

Hence the dual problem $\max_y L(x,y)$ can be computed separately, in parallel:

- Find $x_1^{k+1}, \dots, x_N^{k+1}$ in parallel: $x_i^{k+1} = \operatorname{argmin} L_i(x_i, y^k)$
- Follow ∇m: $y^{k+1} = y^k + s_k (Ax^{k+1} - b)$

↝ Time savings, but one issue: $\|\cdot\|_1$ separable but not strictly convex. Need to stabilize.

③ **Augmented Lagrangian:** add penalty term

$$L_\rho(x,y) = f(x) + y^T(Ax-b) + \frac{1}{2}\rho\|Ax-b\|_2^2$$

↝ Now, problem is strictly convex but not separable. Typically allows to choose $s_k = \rho$.

④ **Splitting: Alternating Direction Method of Multipliers.** (ADMM)

Main idea: suppose f can be split in two parts, possibly ℓ^1 and ℓ^2: $f(x) + g(x)$.

We create a new variable z with constraint $x - z = 0$
↝ new list of constraints $Ax + Bz = c$.

$$L_\rho(x,y,z) = f(x) + g(z) + y^T(Ax+Bz-c) + \frac{1}{2}\rho\|Ax+Bz-c\|_2^2.$$

Advantage: x, z can be updated sequentially.
→ Alternance of minimization of f, g:

ADMM

$$x_{k+1} = \arg\min_x L_\rho(x, y_k, z_k)$$

$$z_{k+1} = \arg\min_z L_\rho(x_{k+1}, y_k, z)$$

$$y_{k+1} = y_k + \rho(Ax_{k+1} + Bz_{k+1} - c)$$

Typically, slow to converge but fast to acceptable.

Tuesday, April 26

Applications of ADMM

Recall problem: minimize $f(x)+g(x)$ subject to $Ax=b$

Form **augmented Lagrangian**:

$$L_\rho(x,y,z) = f(x)+g(z)+y^T(\tilde{A}x+Bz-\tilde{b}) + \rho/2 \,\|\tilde{A}x+Bz-\tilde{b}\|_2^2$$

with $\tilde{A}=\begin{bmatrix}A\\ I\end{bmatrix}$, $B=\begin{bmatrix}0\\ -I\end{bmatrix}$, $\tilde{b}=\begin{bmatrix}b\\ 0\end{bmatrix}$

ADMM

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \; L_\rho(x, y_k, z_k)\\ z_{k+1} &= \operatorname*{argmin}_z \; L_\rho(x_{k+1}, y_k, z)\\ y_{k+1} &= y_k + \rho(Ax_{k+1}+Bz_{k+1}-c) \end{aligned}$$

Last tweak: scaling $u := y/\rho$, combine linear/quadratic

At step k:

$$\begin{cases} x_{k+1} = \operatorname*{argmin}_x \; f(x) + \frac{1}{2}\rho\,\|Ax+Bz_k-c+u_k\|_2^2\\ z_{k+1} = \operatorname*{argmin}_z \; g(z) + \frac{1}{2}\rho\,\|Ax_{k+1}+Bz-c+u_k\|_2^2\\ u_{k+1} = u_k + Ax_{k+1}+Bz_{k+1}-c \end{cases}$$

Examples.

(a) Classical problem: $\min_{x\in K} f(x)$.

Let $g(x)=\begin{cases}0, & x\in K\\ +\infty, & \text{otherwise}\end{cases}$ and use ADMM on

Minimize $f(x)+g(z)$ subject to $x-z=0$

⇝ Scaled Lagrangian:

$$L_\rho(x, z, u) = f(x) + g(z) + \frac{1}{2}\rho \|x - z - u\|^2$$

Notice how usual $\lambda^T(x-z)$ term folded into quadratic term.

⇝ ADMM

$$\begin{cases} x_{k+1} = \operatorname{argmin}_x \left[f(x) + \frac{1}{2}\rho \|x - z_k + u_k\|^2 \right] \\ z_{k+1} = \text{projection of } x_{k+1} + u_{k+1} \text{ onto } K \\ u_{k+1} = u_k + x_{k+1} - z_{k+1} \end{cases}$$

Algorithm alternates minimization and projection.

(b) Soft thresholding. Consider

$$f(x) = \lambda \|x\|_1 = \lambda |x_1| + \dots + \lambda |x_n|$$
$$\rightsquigarrow f_i(x_i) = \lambda |x_i|.$$

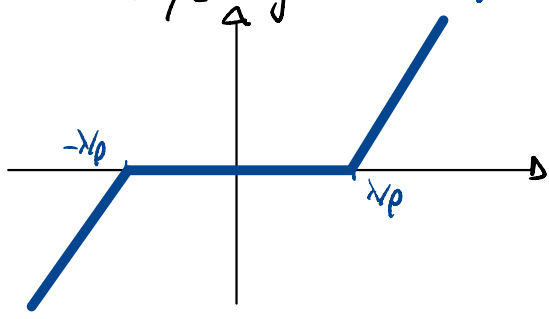
Splitting by ADMM leads to minimization of scalar functions,

$$f_i(x_i) = \lambda |x_i| + \frac{1}{2}\rho (x_i - v_i)^2$$

with $v_i = z_i - u_i$. Solution is "soft thresholding",

$$x_i^* = \left(v_i - \frac{\lambda}{\rho} \right)_+ \quad \text{with} \quad t_+ = \begin{cases} 0, & t \le 0 \\ t, & t \ge 0 \end{cases}$$

$\rightsquigarrow$ Shrinkage of each v_i to zero.



$\Rightarrow$ Problem: $\min_x f(x) + \lambda \|x\|_1$

ADMM:
$$\begin{cases} x_k = \operatorname{argmin}_x f(x) + \frac{1}{2}\rho \|x - z_k + u_k\|_2^2 \\ z_{k+1}^i = \left(x_{k+1}^i + u_k^i - \lambda/\rho\right)_+ \quad \text{(SHRINK)} \\ u_{k+1} = u_k + x_{k+1} - z_{k+1} \end{cases}$$

Other idea: **Proximal Gradient Descent.**

Given $F(x)$, convex:
$$\operatorname{Prox}_{F,\rho}(v) := \operatorname{argmin}_x \left[F(x) + \frac{\rho}{2}\|x - v\|_2^2 \right]$$

Then $\min_x f(x) + g(x)$ computed with

$$x_{k+1} = \operatorname{Prox}_{g,\rho}\left(x_k - s_k \nabla g(x_k)\right)$$

τ_ρ ?

$\Rightarrow$ Useful when $\operatorname{Prox}_{F,\rho}$ known analytically.

c) LASSO

Minimize $\frac{1}{2}\|Ax-b\|_2^2 + \lambda \|x\|_1$

We split the problem into $f(x)+g(z)$, $x-z=0$.

ADMM
- $x_{k+1} = \arg\min \frac{1}{2}\|Ax-b\|_2^2 + \frac{\rho}{2}\|x - z_k + u_k\|_2^2$
  - LS: $x_{k+1} = (A^TA+\rho I)^{-1}(A^Tb + \rho(z_k-u_k))$
- $z_{k+1} = \arg\min \lambda\|z\|_1 + \frac{\rho}{2}\|x_{k+1} - z + u_k\|_2^2$
  - Shrink: $z^i_{k+1} = (x^i_{k+1} + u_k - \frac{\lambda}{\rho})_+$
- $u_{k+1} = u_k + x_{k+1} - z_{k+1}$

d) Compressed sensing; Matrix completion.

Idea: signal to be represented sparsely using basis $v_1, \dots, v_n$

"Sensed" in different basis $w_1, \dots, w_n$

Ex: v's Fourier basis, w's are spike (pixel) basis.
V = Fourier matrix, W = Identity.

Requirement: "incoherence", $V^TW = (v_i^T w_j)$ small (equal size).

Sensing step: measure $y = W^T f$ (unknown signal)

$m < n$ coefficients

$\rightsquigarrow$ minimize $\|x\|_1$ subject to $W^TVx = y$.

Theorem (Candès, Tao, Donoho) Suppose V, W incoherent and x^* sparse. Then with overwhelming probability, when $m > CS\log n$, solution reproduces f exactly.

LASSO with noise: minimize $\|x\|_1$ subject to $\|Ax-b\|_2 \le \varepsilon$.

Netflix problem

Ratings matrix

viewers

$$A_0 = \begin{bmatrix} \times & \times & & \\ & \times & & \times \end{bmatrix} \quad \text{films}$$

few known entries

Completion problem: nuclear norm $\|A\|_N = \sigma_1 + \ldots + \sigma_r$

Minimize $\|A\|_N$ subject to $A = A_0$ in known entries.

Mathematically, proof that A can likely be recovered if enough entries are known ($K > C n^{5/4} r \log n$)

Many algorithms to find optimal solution (10^{10} entries!)

Sparse + low rank decomposition:

Minimize $\|L\|_N + \lambda \|S\|_1$ subject to $L + S = M$.

↑ low-rank ↑ sparse

Application: e.g. surveillance video analysis.

data matrix $\propto$ time vs pixels

↳ low-rank: background, not moving
sparse: characteristic features in particular frames (e.g. a person)

Solution by ADMM:

$$\begin{cases} L_{k+1} = S^{tr}_{1/\rho}\left(\Pi - S_k + W_k\right) \\ S_{k+1} = S^{\ell^1}_{\lambda/\rho}\left(\Pi - L_{k+1} + W_k\right) \\ W_{k+1} = W_k + \Pi - L_{k+1} - S_{k+1} \end{cases}$$

with soft-thresholding operators,

$$S^{tr}_{1/\rho}(A) = U\left(\Sigma - 1/\rho\right)_+ V^T \quad \text{where} \quad A = U\Sigma V^T$$

(soft-thresholding of singular values)

$$S^{\ell^1}_{\lambda/\rho}(A) = \left[\left(a_{ij} - \frac{\lambda}{\rho}\right)_+\right]_{ij}$$

(element-wise soft thresholding)

Thursday, April 28

Gradient Descent

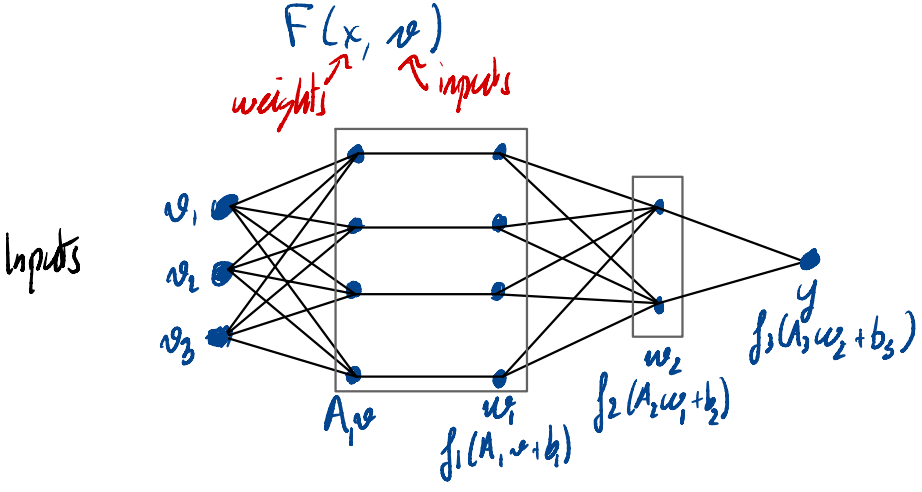
Problem: training models from machine learning often

$$\min_{x} L(x)$$

x: parameters; $L(x)$: loss function

Example: Neural networks:

$F(x, v)$: x = weights, v = inputs



Weights: matrix entries A_1, A_2, A_3; vector entries b_1, b_2, b_3 } x

Given training samples $\{v_i, y_i\}$

v_i: entry data; y_i: output data

Functions f_i typically nonlinear:

e.g. $\mathrm{RELU}(x) = \begin{cases} 0, & t \le 0 \\ t, & t \ge 0 \end{cases}$

Denote $\hat{y}_i = F(x, w_i)$

① Square loss: $L(x) = \frac{1}{N} \sum_{i=1}^{N} \| \hat{y}_i - y_i \|^2$

② Hinge loss: $L(x) = \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i \hat{y}_i)$

for classification $y_i = -1, 1$

③ Cross-entropy loss

$$L(x) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \right]$$

with labels $y_i = 0, 1$ (logistic regression).

Often, many more weights than data points.

Key points:

- Large-scale minimization
- Divide data into training and test data to evaluate "generalization"

- Not interested in "exact" minimum / perfect fit,
  e.g $F(x, v_i) = y_i$.
  ↳ usually badly generalizes on new data v.
- Usually highly non-convex.

I) Classical gradient descent.

Given problem: Find $x^* = \operatorname{argmin} F(x)$
Follow gradient downhill:

Given x_k, $x_{k+1} = x_k - s_k \nabla f(x_k)$
(s_k = step size)

"Steepest descent"

Observation: given quadratic problem $F(x) = \frac{1}{2}(ax^2 + by^2)$.

$$\nabla F(x_0, y_0) = \begin{bmatrix} ax_0 \\ by_0 \end{bmatrix}$$

Steepest line through (x_0, y_0) has equation

$$\begin{vmatrix} x - x_0 & ax_0 \\ y - y_0 & by_0 \end{vmatrix} = 0 \iff ax_0(y - y_0) = by_0(x - x_0)$$

⇝ Lowest point $(0,0)$ is not on this line unless $a=b$.
⇝ need to change directions as we go down.

Classical solution: adjust step size s_k with backtracking.

Zig-zag problem: assume $a=1$, $0<b\leq 1$.

$$f(x,y) = \frac{1}{2}(x^2+by^2). \qquad \nabla f(x,y) = \begin{bmatrix} x \\ by \end{bmatrix}.$$

Assuming perfect line search from $(x_0,y_0) = (b,1)$

we find $$x_1 = \begin{bmatrix} b \\ 1 \end{bmatrix} - s_1 \begin{bmatrix} b \\ b \end{bmatrix}$$

$$f(x_1) = \frac{1}{2}\left((1-s_1)^2 b^2 + b(1-s_1 b)^2\right)$$

$$\frac{df(x_1)}{ds_1} = (1-s_1)b^2 + (1-s_1 b)b^2$$

$$= b^2(2 - s_1(1+b)) = 0$$

$$s_1 = \frac{2}{1+b}$$

hence $$x_1 = \begin{bmatrix} b - \frac{2b}{1+b} \\ 1 - \frac{2b}{1+b} \end{bmatrix} = \frac{1}{1+b}\begin{bmatrix} b(1+b) - 2b \\ 1+b-2b \end{bmatrix} = \frac{1-b}{1+b}\begin{bmatrix} -b \\ 1 \end{bmatrix}$$

By recurrence, we find $$x_k = b\left(\frac{b-1}{b+1}\right)^k \qquad y_k = \left(\frac{1-b}{1+b}\right)^k$$

and $f(x_{k+1}, y_{k+1}) = \left(\frac{1-b}{1+b}\right)^{2k} f(x_0, y_0)$

- If $b \approx 1$, solution almost immediately goes to $(0,0)$. Works great.
- If $b \approx 0$ (ill-conditioned S) slow progress since $\frac{b-1}{b+1} \not\approx 1$.

Convergence Analysis. Given $F(x)$ smooth, strongly convex, $H(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]$ has eigenvalues $m \le \lambda \le M$ $\forall x$.

Now, recall $x_{k+1} = x_k - s_k \nabla f(x_k)$,

$$F(x_{k+1}) \le F(x_k) + \nabla f(x_k)^T (x_{k+1} - x_k) + \frac{M}{2} \|x_{k+1} - x_k\|^2$$

$$= F(x_k) - s\|\nabla F(x_k)\|^2 + \frac{Ms^2}{2} \|\nabla F(x_k)\|^2.$$

Minimizing over s yields $F(x_{k+1}) \le F(x_k) - \frac{1}{2M} \|\nabla f(x_k)\|^2$.

Also,

$$F(x^*) \ge F(x_k) - \frac{1}{2m} \|\nabla F(x_k)\|^2.$$

Hence $F(x_{k+1}) - F(x^*) \le \left(1 - \frac{m}{M}\right)\left(F(x_k) - F(x^*)\right)$

⇝ Slow convergence when $m/n \ll 1$.

Remark: In practice, no exact line search but backtracking.

II) Momentum, Heavy ball

Main idea: a heavy ball rolling downhill does not zig-zag. Keeps momentum.

$$x_{k+1} = x_k - s z_k \quad \text{with} \quad z_k = \nabla F(x_k) + \beta z_{k-1}$$

Two coefficients: s, step size
β, momentum decay.

Descent with momentum:

$$\begin{cases} x_{k+1} = x_k - s z_k \\ z_{k+1} = \nabla f(x_{k+1}) + \beta z_k \end{cases}$$

One-step formulation.

LD looks like 2nd order equation,

$$\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky = 0. \iff \frac{d}{dt}\begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k & -b \end{bmatrix}\begin{bmatrix} y \\ y' \end{bmatrix}$$

Choosing s, β Quadratic model $F(x) = \frac{1}{2} x^T S x$:

optimal choice $s = \left(\frac{2}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}}\right)^2$, $\beta = \left(\frac{\sqrt{\lambda_{max}} - \sqrt{\lambda_{min}}}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}}\right)^2$

Example $S = \begin{bmatrix} 1 & 0 \\ 0 & b \end{bmatrix}$ then $s = \left(\frac{2}{1+\sqrt{b}}\right)^2$, $\beta = \left(\frac{1-\sqrt{b}}{1+\sqrt{b}}\right)^2$

Descent factor $\sim \left(\frac{1-\sqrt{b}}{1+\sqrt{b}}\right)^2$ from $\left(\frac{1-b}{1+b}\right)^2$.

So much improved if $b \approx 0$. Take $b = 0.01$,

Steepest descent: $\left(\frac{.99}{1.01}\right)^2 \approx 0.96$,

Accelerated: $\left(\frac{.9}{1.1}\right)^2 \approx 0.67$

Note $\lambda_{max}/\lambda_{min} = 1/b = \kappa$ condition number of S.

III Nesterov acceleration.

Last idea: evaluate ∇f at $x_k + \gamma_k (x_k - x_{k-1})$.

So $\begin{cases} x_{k+1} = y_k - s \nabla f(y_k), \\ y_{k+1} = x_{k+1} + \beta (x_{k+1} - x_k). \end{cases}$

Same improvement as before in analysis – much improved rate in practice.

IV Stochastic Gradient Descent.

Applies to machine learning where

$$L(x) = \frac{1}{N} \sum_{i=1}^{N} \ell(x, w_i, y_i)$$

$\Rightarrow$ computing gradient ∇L is expensive.

Observe $\nabla L(x) = \mathbb{E}[\nabla \ell(x, w_i, y_i)]$

where i random index.

$\Rightarrow$ Cheap estimate of ∇P!

Stochastic Descent with one sample per step

$$x_{k+1} = x_k - \nabla_x \ell(x_k, w_{i_k}, y_{i_k})$$

"random" index chosen for each step.
Usually, random ordering of samples.

Stochastic Descent with B samples per step

$$x_{k+1} = x_k - \frac{1}{B} \sum_{i \in I_k} \ell(x_k, w_i, y_i)$$

Random mini-batch
Set of B indices / samples

Epoch: one complete pass through training data.

Behavior of algorithm:

- Fast convergence at first;
- Large oscillations near solution.

Stopping early avoids overfitting.

II) Adaptive Stochastic Gradient

Main idea: form combination from previous steps,

$$D_k = D(\nabla L_k, \nabla L_{k-1}, \dots)$$

$$s_k = s(\nabla L_k, \nabla L_{k-1}, \dots)$$

↑ computed from mini-batches.

ADAGRAD: adjust step-size,

$$D_k = L_k, \quad s_k = \left(\frac{\alpha}{\sqrt{k}}\right)\left[\frac{1}{k}\,\mathrm{diag}\left(\sum_{i=1}^{k} \|\nabla L_i\|^2\right)\right]^{1/2}$$

ADAD: adjust gradients and step sizes,

$$D_k = (1-\vartheta)\sum_{i=1}^{k} \vartheta^{k-1}\nabla L(x_i),$$

$$s_k = \left(\frac{\alpha}{\sqrt{k}}\right)\left[(1-\beta)\,\mathrm{diag}\sum_{i=1}^{k}\beta^{k-1}\|\nabla L_i\|^2\right]^{1/2}$$

$\rightsquigarrow$ Exponential moving averages.

Typically: $\delta = 0.9$, $\beta = 0.999$.

Practical formulae:

$$\begin{cases} D_k = \delta D_{k-1} + (1-\delta) \nabla L(x_k) \\ s_k^2 = \beta s_{k-1}^2 + (1-\beta) \|\nabla L(x_k)\|^2 \end{cases}$$

Very popular!