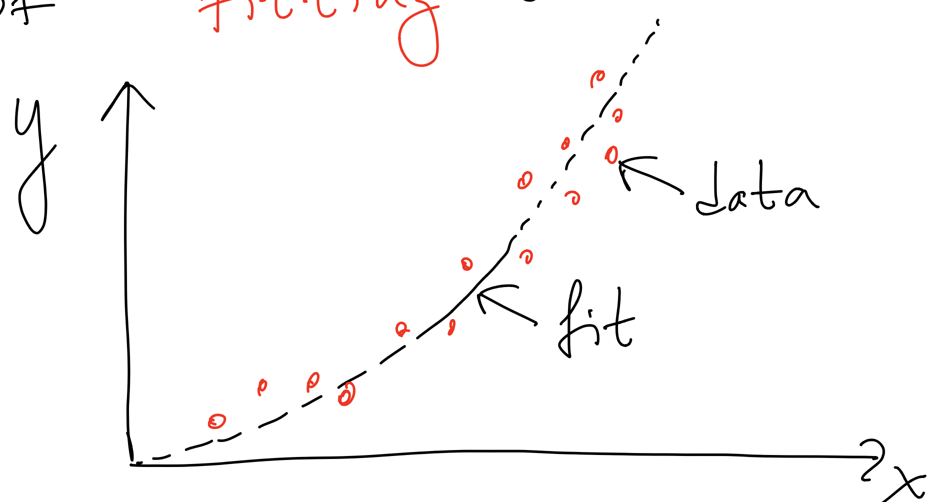
Overdetermined Linear Systems of Equations

A. DONEV, Spring 2021

To motivate the problem, let's consider the problem of fitting or linear regression



Data: (x_1, y_1)
(x_2, y_2)
$\vdots$

Model: $y(x) = a + bx + cx^2$

1

But the real data has errors / perturbations or our model is not perfect

$$y_i = a + b x_i + c x_i^2 + \varepsilon_i$$

"noise" or error

We want to find a, b, c i.e. find the best fit

Residual or error in fit

$$r_i(a,b,c) = y_i - (a + b x_i + c x_i^2)$$

"Best fit" is one that minimizes the residual.

$$(a,b,c) = \arg\min_{a,b,c} \|\vec{r}\|$$

We need to choose the norm. The easiest choice is L_2: (linear) least squares fitting.

Matrix-vector notation

$$\vec{y}_{model}(\vec{p}) = \overleftrightarrow{X}\,\vec{p}$$

observations, data, parameters

$$\overrightarrow{X} = \left[\begin{array}{c|c|c} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ \vdots & \vdots & \vdots \end{array}\right], \quad \vec{p} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\vec{p} = \arg \min_{\vec{p}} \| \overleftrightarrow{X} \vec{p} - \vec{y} \|_2$$

or the same

$$p = \arg \min_{p} \| Xp - y \|_2^2$$

This problem is often written as an overdetermined linear system

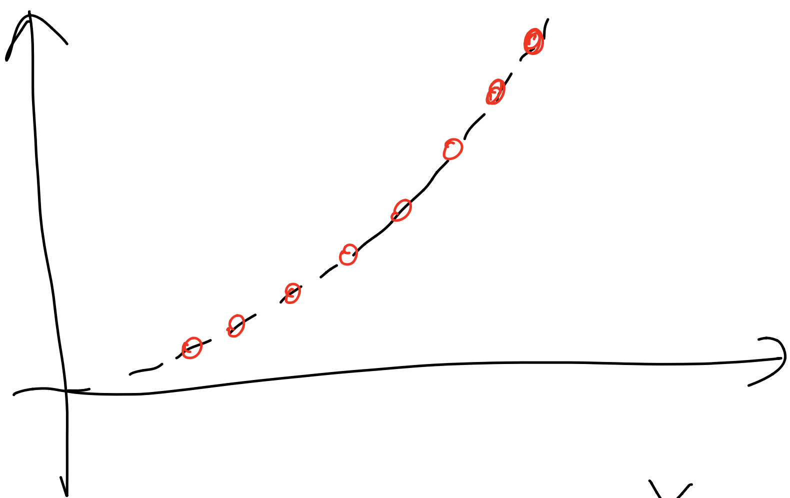
$$y = X p$$

(more equations than unknowns) but this is just notation.

In Matlab

$p = X \backslash y$ works

If there were no errors



then indeed $y = Xp$ is satisfied but there are lots of redundant equations (we only need 3 points to fit a parabola)

For consistency with other sources & previous lecture

$$Ax = b$$

$$A = [m, n], \quad m >> n$$

$Ax = b$, A not square

Are there solutions?

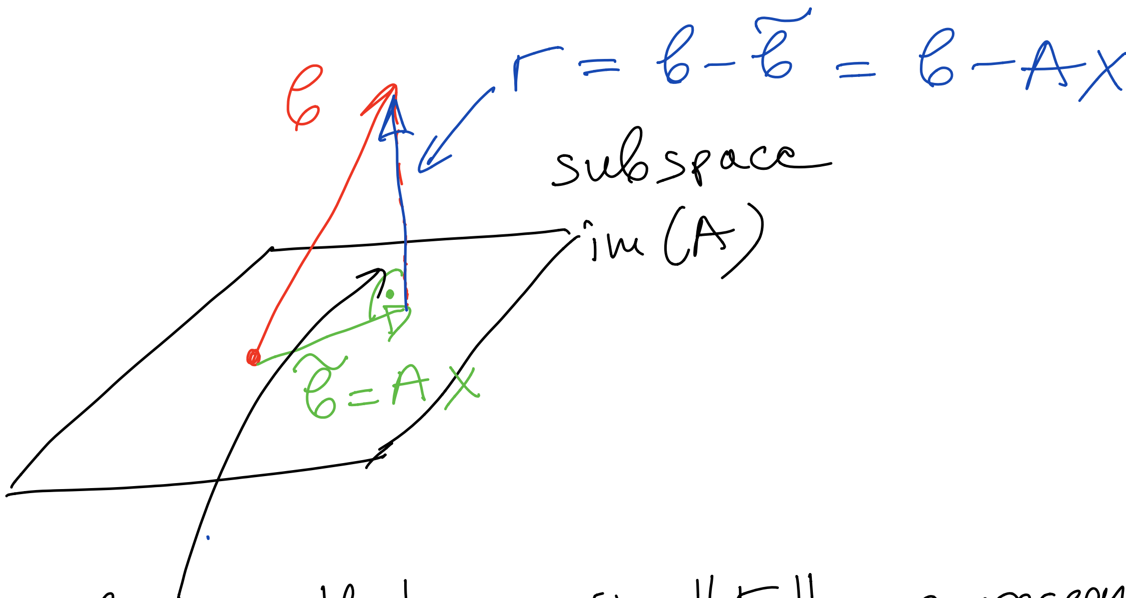
If $b \in \text{im}(A)$

column space or image

then there is at least one solution. But if $b \notin \text{im}(A)$, then we can project it onto $\text{im}(A)$ and solve

$$Ax = \text{proj}_{\text{im}(A)} b = \tilde{b}$$

and now this can be solved, for example, by choosing any n of the m rows (linearly independent)



Observe that $\min \|r\|_2$ corresponds to orthogonal projection (also called L_2 projection)

$$\vec{r} \perp \mathrm{im}(A)$$

$$(b - Ax) \perp \mathrm{im}(A)$$

$$\Rightarrow \quad a_i \cdot (b - Ax) = 0 \ \forall i$$

a_i: column of matrix A = row of matrix A^T

$\Rightarrow$ In matrix notation

$$A^T(b - Ax) = 0$$

$$\Rightarrow \quad (A^T A)x = A^T b$$

normal equations

It looks like all we did was just multiply $Ax = b$ by A^T from the left

$$A^T = [n \times m]$$
$$A = [m \times n]$$
$$\Rightarrow \quad A^T A = [n \times n]$$
$$A^T b = [n \times m][m \times 1] = [n \times 1]$$

Normal equations are a square symmetric linear system

Aside:

Matrix $A^T A$ is also positive definite (all eigenvalues are positive) so Matlab will use Cholesky factorization instead of LU,

$$A^T A = L L^T$$

L ↑ lower triangular

Computational cost

Compute $B = A^T A$:

$$[n \times m]\,[m \times n] =$$

$$= n^2 \cdot m \quad \text{FLOPS}$$

↑ dot product

Solve $Bx = A^T b$ $= O(n^3)$ FLOPS

But since $m >> n$,

$$mn^2 >> n^3$$

So the cost is $O(mn^2)$ FLOPS

This is as good as any <u>exact</u> algorithm.

But Matlab does <u>not</u> use normal equations for $A \backslash b$.

Main reason is ill-conditioning

$$K_2(A^T A) = (K_2(A))^2$$

So if A is ill-conditioned (e.g. same x but multiple values of y when fitting) we run into problems

Another idea :

Find an orthonormal basis for im (A)

$$\{\vec{q}_1, \ldots, \vec{q}_n\} \Longleftrightarrow Q$$

(assume here A is full rank)

$$q_i \cdot q_j = \begin{cases} 1 & \text{if } i = j \\ \emptyset & \text{if } i \neq j \end{cases}$$

$$Q^T Q = I \leftarrow \text{identity matrix}$$

orthogonal matrix

$$\begin{cases} \operatorname{im}(Q) = \operatorname{im}(A) \\ Q^T Q = A \end{cases}$$

Q <u>not</u> unique! How to find one Q ?

Answer: Gram-Schmidt (GS) process

Given vectors

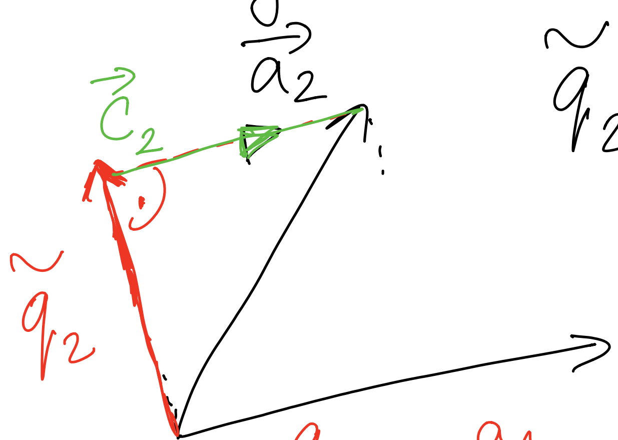
$\{a_1, \dots, a_n\}$

produce orthogonal basis vectors

$\{\tilde{q}_1, \dots, \tilde{q}_n\}$

with the same span

Easy in 2D:

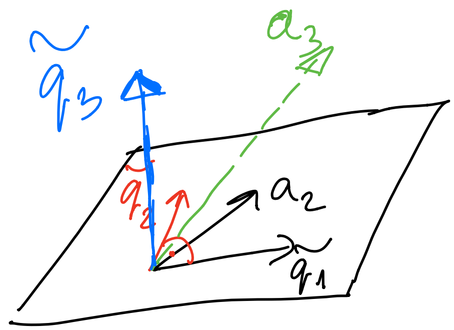


$$\tilde{q}_2 = \vec{a}_2 - \vec{c}_2 =$$
$$= \vec{a}_2 - (\vec{a}_2 \cdot \tilde{q}_1)\, \tilde{q}_1$$

$$q_1 = \frac{a_1}{\|a_1\|_2}$$

$$q_2 = \frac{\tilde{q}_2}{\|\tilde{q}_2\|_2}$$

3D:



$$\tilde{q}_3 = a_3 - (a_3, q_2)\, q_2 - (a_3, q_1)\, q_1$$

$$q_3 = \frac{\tilde{q}_3}{\|\tilde{q}_3\|_2} \quad \text{(normalize)}$$

Note: Lookup "modified GS" in Wiki

(Standard) GS method: $k = 2, \dots, n$

$$\tilde{q}_{k+1} = a_{k+1} - \sum_{j=1}^{k} (a_{k+1} \cdot q_j)\, q_j$$

$$q_{k+1} = \tilde{q}_{k+1} / \|\tilde{q}_{k+1}\|$$

As we do this, let's save some of the coefficients

$$r_{11} = \|a_1\|_2$$

$$r_{12} = (a_2, q_1)$$

$$r_{22} = \|a_2 - r_{12} q_1\|$$

all quantities computed during GS, we just need to do book keeping and store them

Put in upper triangular matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & \ddots & & \vdots \\ \emptyset & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix}$$

And, it turns out that, like for LU factorization via GEM

$$A = QR$$ QR factorization

orthogonal matrix — upper triangular matrix

$$[m \times n] = [m \times n][n \times n]$$

$$Q^T Q = I \Rightarrow$$

$$Q^{-1} = Q^T \text{ if } m = n$$

The QR factorization is just as useful (and <u>more</u> useful) than an LU factorization

So if A is square,

$$A^{-1} = (QR)^{-1} = R^{-1} Q^{-1}$$

$$A^{-1} = R^{-1} Q^{T}$$

$$\Rightarrow \quad x = A^{-1} b = R^{-1} Q^{T} b$$

Rewrite as

$$\begin{cases} y = Q^{T} b \\ \text{solve} \quad R x = y \end{cases}$$

← upper triangular system! (forward subst)

Now, it turns out that this works even for overdetermined linear systems

$Q^{T} = [n \times m] \quad b = [m \times 1]$

$Q^T b = [n \times 1]$

$R = [n \times n]$

so works!

Proof:

Go back to normal equations

$$(A^T A)x = A^T b$$

$$A = QR$$

$$(R^T \underbrace{Q^T Q}_{I} R)x = R^T Q^T b$$

$$R^{-T} \backslash \quad R^T (Rx) = R^T Q^T b$$

$$\Rightarrow \quad Rx = Q^T b$$