Lecture 14: QR decomposition

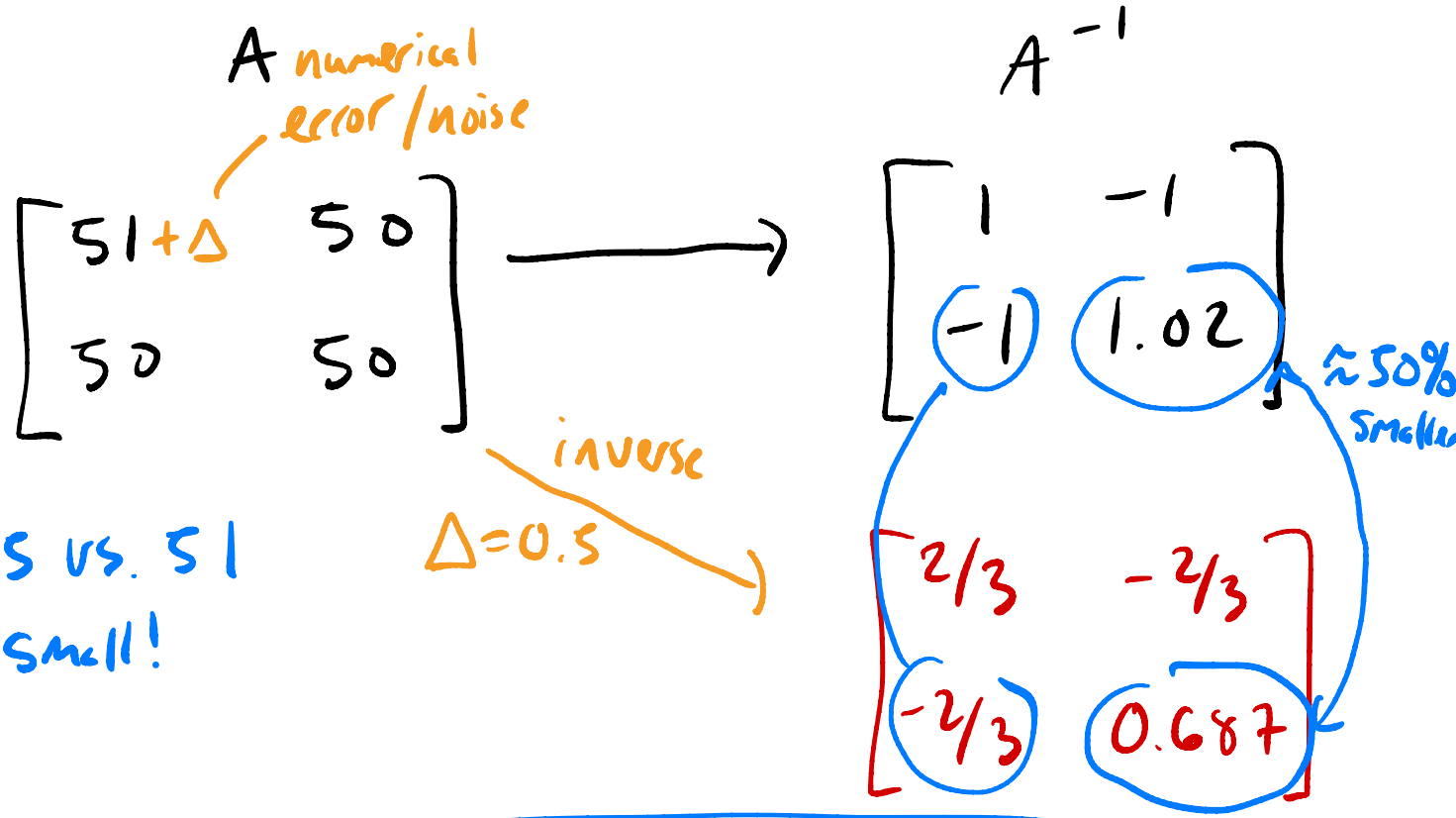
Normal eqs: Solve $A\vec{c} = \vec{y}$

$$\vec{c} = (A^T A)^{-1} A^T \vec{y}$$

Problems

1) Computationally intense — inverse is costly.

2) Inverse operation is sensitive to noise/numeric errors.



Want to avoid taking inverses whenever possible!

Alternative : matrix factorization.

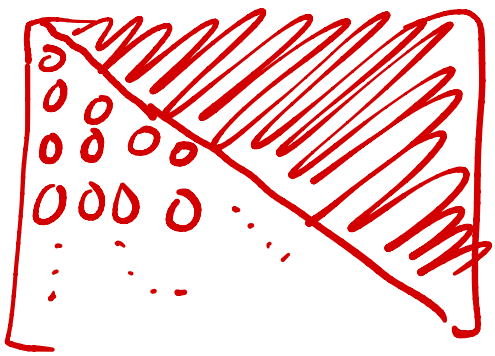
$$\underset{(N,\,M+1)}{A} = \underset{(N,\,M+1)}{Q}\ \underset{(M+1,\,M+1)}{R}$$

Q orthogonal matrix : columns of Q are mutually orthogonal vectors, length 1 (unit vectors)

$$Q^T Q = I \Rightarrow Q^T = Q^{-1}$$

$$Q^T Q Q^{-1} = I Q^{-1}$$

R upper triangular matrix



$$\begin{bmatrix} 1 & 2 & & & & \\ 0 & 1 & & & & \\ 0 & 0 & 1 & 4 & 4 & 4 & 6 \\ 0 & 0 & & 1 & 2 & -3 & 9 \\ \vdots & & & & \ddots & & 2 \\ & & & & & & 1 \end{bmatrix}_{M+1}$$

$M+1$

Solving for Q is a process. Gram-Schmidt orthogonalization algorithm

R is easy. Once we know Q, we can solve for R:

$$A = QR$$

$$Q^{-1}A = Q^{-1}QR$$

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

(flip)

$$R = Q^T A$$

$(N, m+1)$

$(m+1, m+1)$

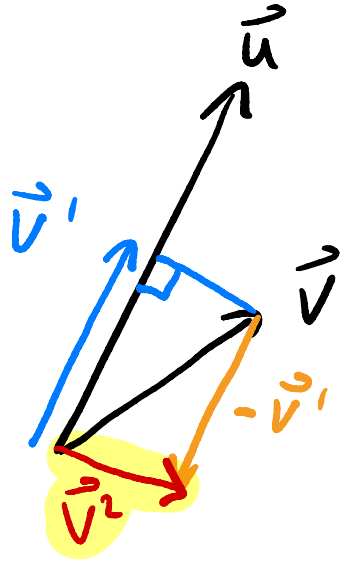
$(m+1, N)$

Gram-Schmidt algorithm (intuition):

Idea: orthogonalize cols of A (data vars) by:

Subjecting off "overlapping" components between pairs of vectors — vector projection.

Example: Want to make $\vec{v}$ orthogonal to $\vec{u}$



Compute projection of $\vec{v}$ onto $\vec{u}$:

$$\text{proj}_{\vec{u}}(\vec{v}) = \vec{v}'$$

Need to still make $\vec{v}$ perpendicular to $\vec{u}$:

$$\vec{v}^2 = \vec{v} - \overbrace{\text{proj}_{\vec{u}}(\vec{v})}^{\vec{v}'} \quad : \quad \vec{v} + (-\vec{v}')$$

$$\underbrace{\qquad\qquad}_{-\vec{v}'}$$

General Case: To make col vectors:

original data vectors $\longrightarrow$ $[\vec{v}_1, \vec{v}_2, \vec{v}_3, \ldots, \vec{v}_m]$ orthogonal:

mutually orthogonal $\longrightarrow$ $[\vec{u}_1, \vec{u}_2, \vec{u}_3, \ldots, \vec{u}_m]$

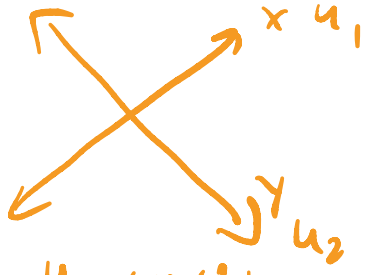
progressively subtract out projections onto previous col vectors:

orthogonal ← original

$$\vec{u}_1 = \vec{v}_1$$

$\vec{v}_2$ above

$$\vec{u}_2 = \vec{v}_2 - \text{proj}_{\vec{u}_1}(\vec{v}_2)$$



u_3 would be pointing out the page.

$$\vec{u}_3 = \vec{v}_3 - \text{proj}_{\vec{u}_1}(\vec{v}_3) - \text{proj}_{\vec{u}_2}(\vec{v}_3)$$

$\vdots$

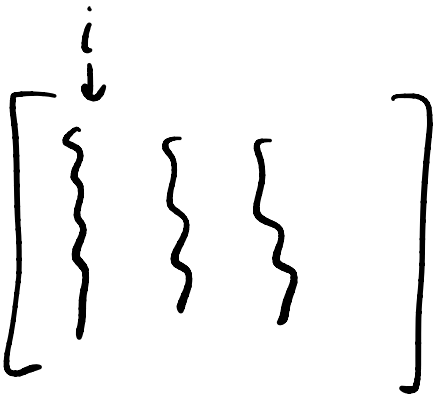
building up mutually orthogonal set of vectors.

Details

1) Initialize Q a $(N, M+1)$ matrix.

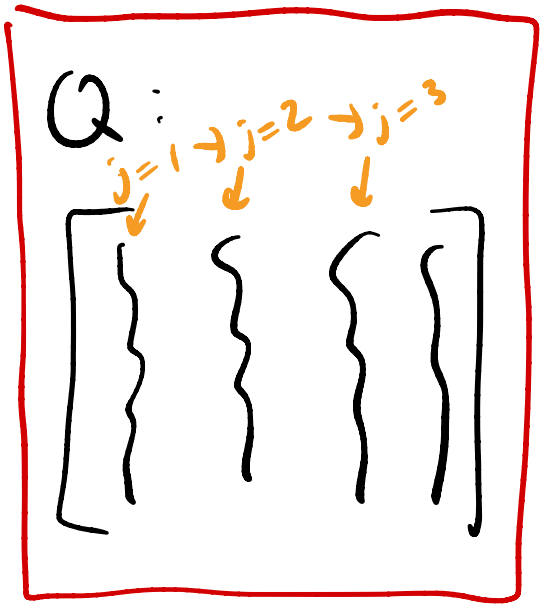
2) Visit cols of A from left → right
index of current col i

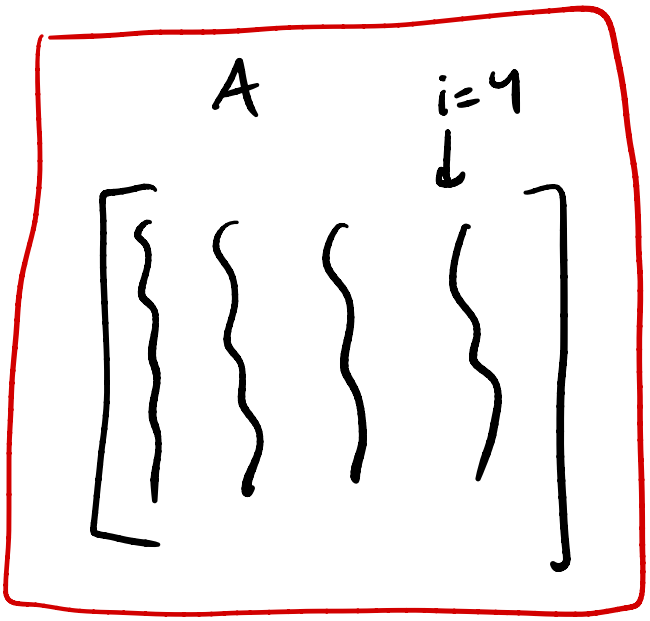
A



3) Define new pointer j, revisit all cols to the left of i in Q, $j < i$

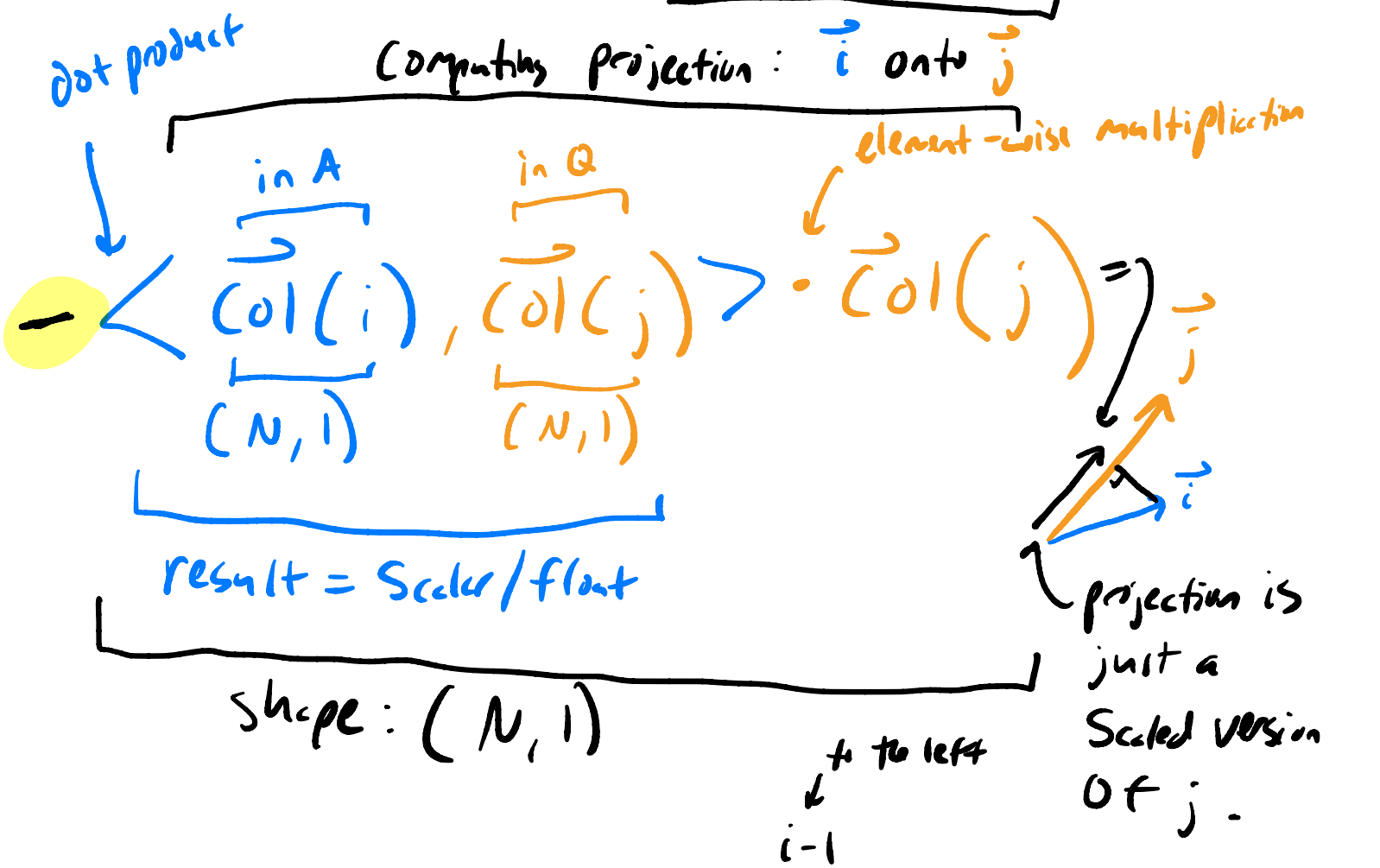
look at cols strictly to the left of i





j purpose is to visit vectors that are already mutually orthogonal.

4) For each valid value of j add the following to a copy of col i of A



ie. $\vec{col}(i) = \vec{col}(i) - \sum_{j=1}^{i-1} \langle \vec{col}(i), \vec{col}(j) \rangle \cdot \vec{col}(j)$

(i-1 : to the left)