

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/numerical errors.

A numerical error/noise

$$\begin{bmatrix} 51+\Delta & 50 \\ 50 & 50 \end{bmatrix} \xrightarrow{\text{inverse}} \begin{bmatrix} 1 & -1 \\ -1 & 1.02 \end{bmatrix}$$

$\Delta = 0.5$

0.5 vs. 51 is small!

$\approx 50\%$ Smaller

$$\begin{bmatrix} 2/3 & -2/3 \\ -2/3 & 0.687 \end{bmatrix}$$

Want to avoid taking inverses whenever possible!

A Iterative : matrix factorization.

$$\tilde{A} = Q \tilde{R}$$

Dimensions:

- \tilde{A} : $(N, M+1)$
- Q : $(N, M+1)$
- \tilde{R} : $(M+1, M+1)$

Q

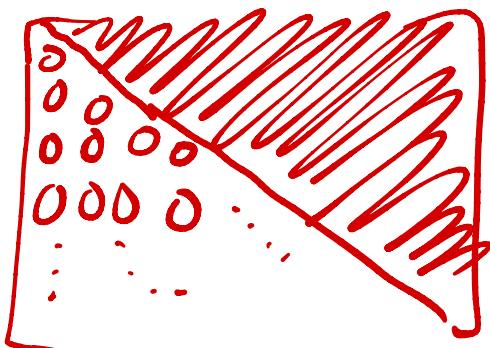
Orthogonal matrix: columns of Q are mutually orthogonal
vectors, length 1
(unit vectors)

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

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

R

upper triangular matrix



$$\begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & \ddots \end{bmatrix}^{M+1} \quad \left[\begin{array}{c|ccccc} 1 & & & & & \\ 0 & 1 & & & & \\ 0 & 0 & 1 & & & \\ 0 & 0 & 0 & 1 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ & & & & & 1 \end{array} \right]^{M+1}$$

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

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

$$A = Q R$$

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

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

$$R = Q^T A$$

Diagram illustrating the dimensions of the matrices in the QR decomposition:

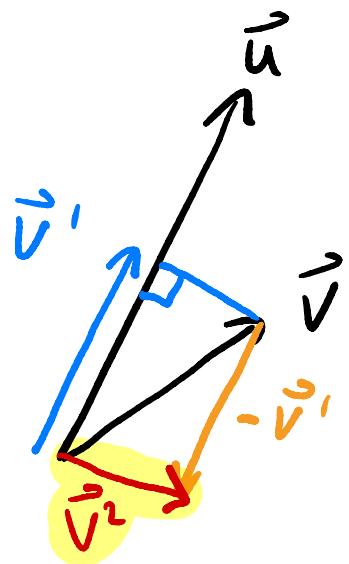
- R : $(N, M+1)$
- Q^T : $(M+1, M+1)$
- A : $(M+1, N)$

(flip)

Gram-Schmidt algorithm (intuition):

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

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

General Case: To make col vectors:

Original data vectors $\rightarrow [\vec{v}_1, \vec{v}_2, \vec{v}_3, \dots, \vec{v}_n]$ orthogonal:

mutually orthogonal $\rightarrow [\vec{u}_1, \vec{u}_2, \vec{u}_3, \dots, \vec{u}_n]$

progressively Subtract Out projections

onto previous col vectors:

Orthogonal

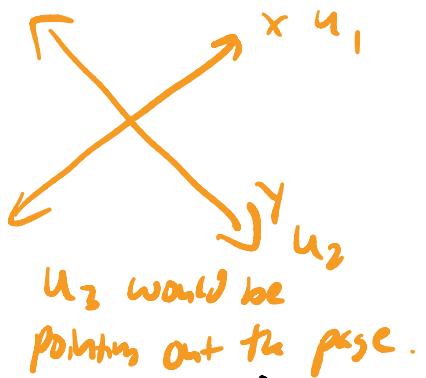
Original

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

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

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

⋮

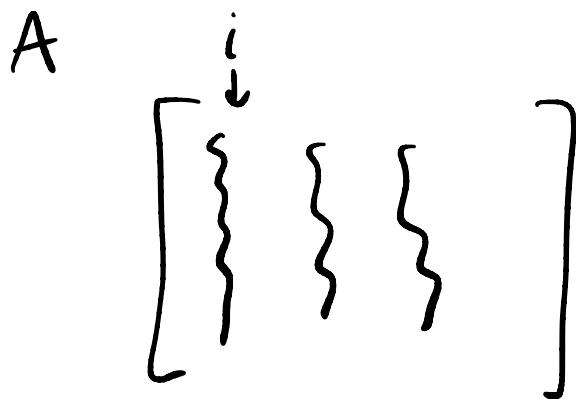


u_3 would be pointing out the page.

building up mutually orthogonal set of vectors.

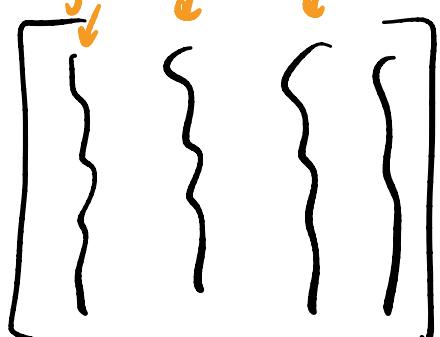
Details

- 1) Initialize Q a $(N, M+1)$ matrix.
- 2) Visit cols of A from left \rightarrow right
index of current col i



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

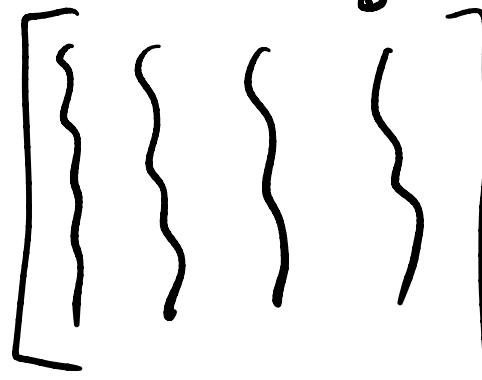
$Q : j=1 \rightarrow j=2 \rightarrow j=3$



j

A

$i=4$



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

dot product

Computing projection: \vec{i} onto \vec{j}

$\vec{\text{col}}(i)$
in A
 $(N, 1)$

$\vec{\text{col}}(j)$
in Q
 $(N, 1)$

element-wise multiplication

$$\langle \vec{\text{col}}(i), \vec{\text{col}}(j) \rangle \cdot \vec{\text{col}}(j) = \vec{j}$$



result = Scalar / float

shape: $(N, 1)$

+ to left
 $i-1$

projection is just a scaled version of j .

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