Lecture 15

Linear regression via QR decomposition.

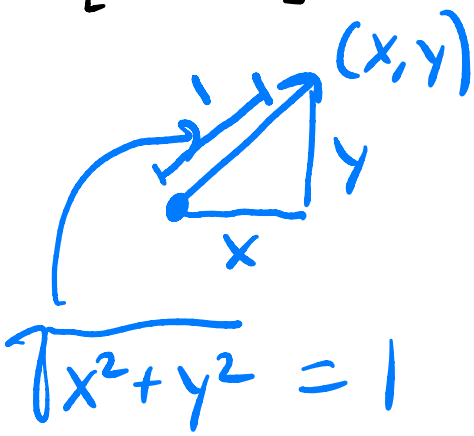
$$\vec{y} = A\vec{c}$$

$(N, M+1)$ $(N, M+1)$ $(M+1, M+1)$

$$A = Q\,R$$

Q: Orthogonal matrix. Cols are mutually perpendicular

[len = 1]

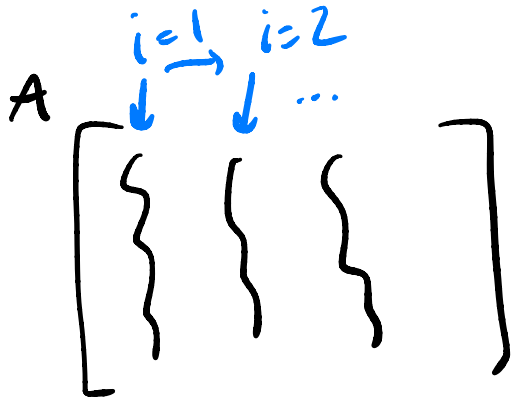


Gram-Schmidt orthogonalizing algorithm:

Goal Compute Q given A.

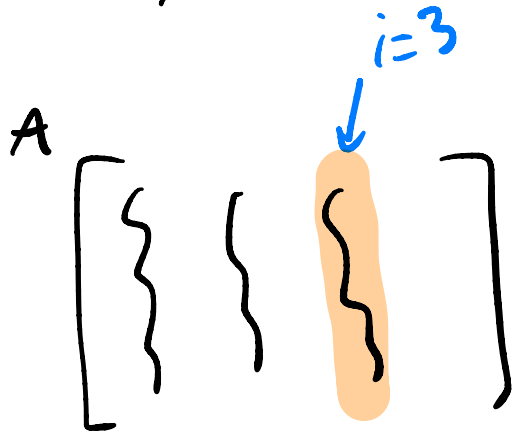
1) Initialize Q : array shape $(N, M+1)$

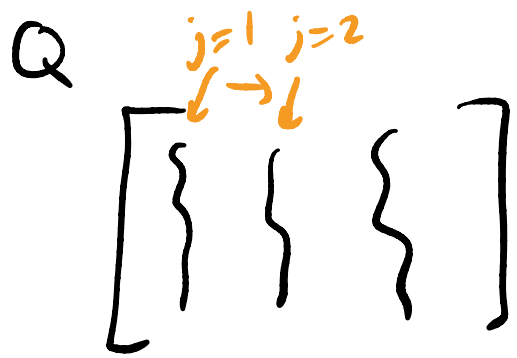
2) Introduce pointer i index columns as we visit them left → right.



3) Introduce another pointer j revisit all Columns to the left of i but in Q not A.

$j < i$



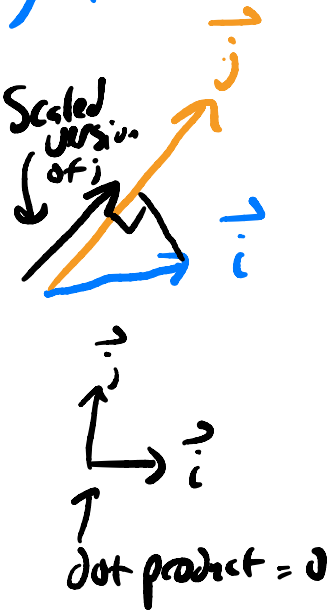


j: accessing set of vectors that are already mutually orthogonal.

4) For all valid j values ($j<i$), add the following to a copy of $\vec{col}(i)$ from A.

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

- $\vec{col}(i)$: from A, $(N,1)$
- $\vec{col}(j)$: from Q, $(N,1)$
- $\langle \cdot, \cdot \rangle$: dot product
- $\cdot$: element-wise multiplication
- result: Scalar
- Scaling the vector $\vec{col}(j)$, $j<i$



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

- $\vec{col}(i)$: Copy of Col i from A — don't change A!

5) Normalize $\boxed{\vec{Col}(i) \text{ copy}}$ by its length / Euclidean distance:

$$\vec{Col}(i) = \frac{\vec{Col}(i)}{\|\vec{Col}(i)\|_2} = \frac{\vec{Col}(i)}{\sqrt{x_{i,1}^2 + x_{i,2}^2 + x_{i,3}^2 + \dots}}$$

$$\vec{Col}(i) = (x_{i,1}, x_{i,2}, x_{i,3}, \dots)$$

6) Assign the $\boxed{\vec{Col}(i) \text{ copy}}$ so that it is now $\boxed{\vec{Col}(i) \text{ of } Q}$

7) Increment $\boxed{i}$. Repeat steps 4-6 with new $\boxed{i}$. Repeat until done with last Col of A.

8) Have Q. Compute $\boxed{R = Q^T A}$

$$A = QR$$

$$Q^{-1}A = Q^{-1}QR \rightarrow Q^{-1}A = R \rightarrow Q^T A = R$$

Have $A = QR$

Solve least squares problem with $A = QR$

$\vec{y} = A\vec{c}$ ← find $\vec{c}$

$\vec{y} = QR\vec{c}$

$Q^T\vec{y} = R\vec{c}$ ← col vector $m+1$ unknowns

$(m+1, n)$ $(n, 1)$ $(m+1, m+1)$

$\Rightarrow (m+1, 1)$

$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{bmatrix}$ $(m+1, 1)$

$\Rightarrow$ flip eq left/right

$R\vec{c} = Q^T\vec{y}$

Multiply out $R\vec{c} \Rightarrow$ look at system of eqs.

$$\text{blah} \cdot c_{n-2} + \cdots$$
$$\vdots$$
$$1 \cdot c_{n-1} + 2 \cdot c_n = 3$$
$$1 \cdot c_n = 2$$

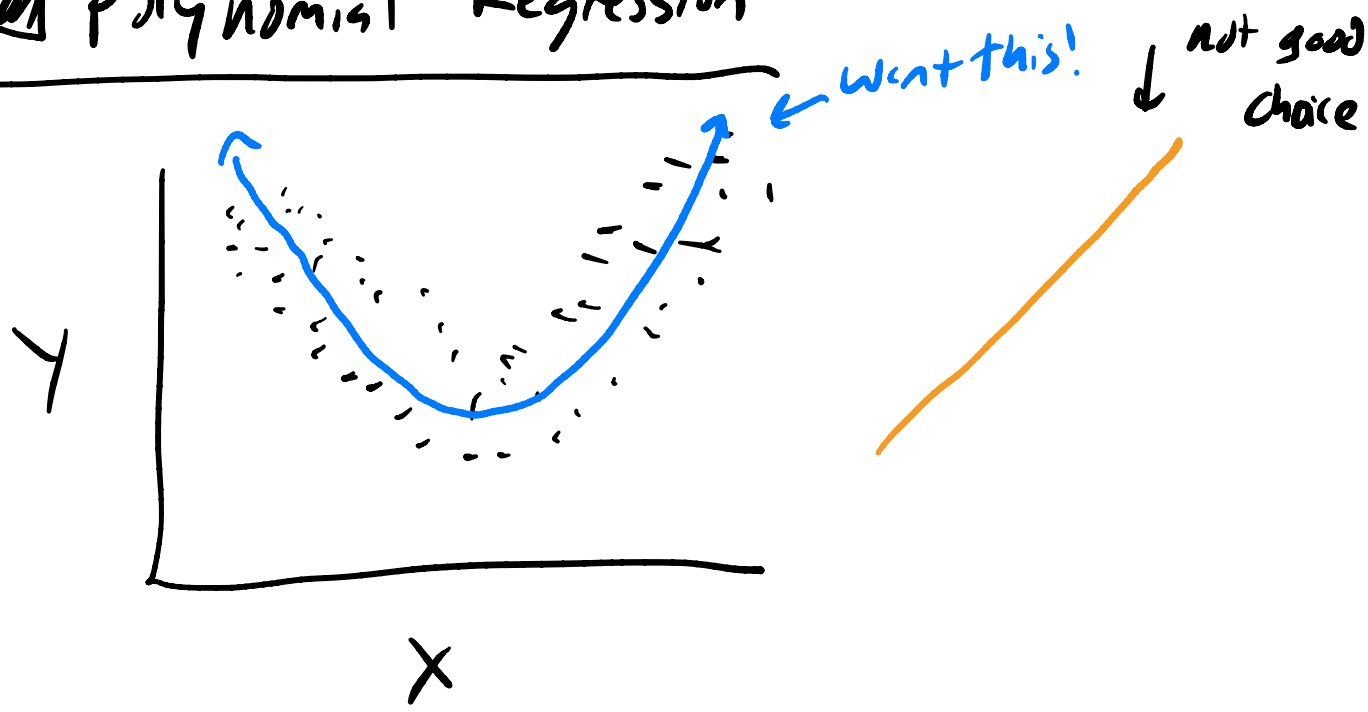
$R\vec{c}$:
$M+1$ equations
w/ $M+1$ unknowns

Solve c_n 1st → can solve for c_{n-1} on line above, etc.

back substitution

`scipy.linalg.solve_triangular` } does back substitution

Polynomial Regression



Example:

$$\vec{X}_1 = \begin{bmatrix} 2 \\ 4 \\ 1 \\ 3 \\ 5 \end{bmatrix} \qquad \vec{Y} \begin{bmatrix} 15 \\ 5 \\ 2 \\ 11 \\ 4 \end{bmatrix}$$

Cubic regression model

$$Y = C_0 + C_1 X_1 + C_2 X_1^2 + C_3 X_1^3$$

Setup $\quad \vec{y} = A\vec{c}$

$$\vec{y} = \begin{bmatrix} 15 \\ 5 \\ 2 \\ 11 \\ 4 \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \\ 1 & 1 & 1 & 1 \\ 1 & 3 & 9 & 27 \\ 1 & 5 & 25 & 125 \end{bmatrix}}^{A} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \vec{c}$$

$x_1^2 \quad x_1^3$

$\Rightarrow$ plug in to solve for $\vec{c}$ using any technique.
normal eqs, QR, Scipy, etc ... Solver of choice.