$X = QR$ decomposition has 2 steps (1) Gram-Shmidt algorithm which converts $X$ into $Q$ column-by-column and (2) reconstruction of the upper triangle change-of-basis matrix $R$. $X$ has ~~d~~imension $n \times k$ and columns $x_1, \dots, x_k$

In 1. we first (a) create a orthogonal basis $v_1, \dots, v_k$ and then (b) normalize its component vectors into $q_1, \dots, q_k$

1. (a) $\vec{v}_1 := \vec{x}_1$

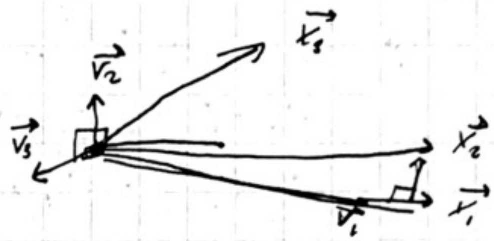
$\vec{v}_2 := \vec{x}_2 - \text{proj}_{\vec{v}_1}(\vec{x}_2)$

$\text{Span}\{\vec{x}_1, \vec{x}_2\} = \text{span}\{\vec{v}_1, \vec{v}_2\}$ but $\vec{v}_1 \perp \vec{v}_2$

$\vec{v}_3 := \vec{x}_3 - \text{proj}_{[\vec{v}_1 | \vec{v}_2]}(\vec{x}_3)$

$\vdots$

$\vec{v}_k := \vec{x}_k - \text{proj}_{[\vec{v}_1 | \dots | \vec{v}_{k-1}]}(\vec{x}_k)$




1(b) $\vec{q}_1 := \dfrac{\vec{v}_1}{\|\vec{v}_1\|}$, $\vec{q}_2 := \dfrac{\vec{v}_2}{\|\vec{v}_2\|}$, $\dots$, $q_k := \dfrac{\vec{v}_k}{\|\vec{v}_k\|}$

$\Rightarrow Q = [\vec{q}_1 | \dots | \vec{q}_k]$

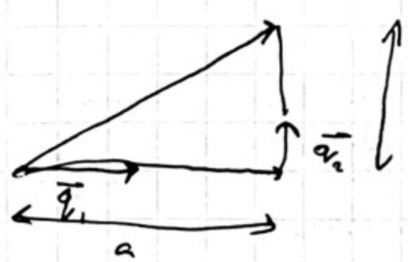
(2) $X = QR$

$$R = \begin{bmatrix} a & b & d & & \text{—} \\ 0 & c & e & & \text{—} \\ 0 & 0 & f & & \text{—} \\ \vdots & \vdots & 0 & \cdots & \text{—} \\ & & \vdots & & \text{—} \\ 0 & 0 & 0 & & \text{—} \end{bmatrix}$$

$[\vec{x}_1 | \vec{x}_2 | \dots | \vec{x}_k] = [\vec{q}_1 | \vec{q}_2 | \dots | \vec{q}_k]$

$\vec{x}_1 = \overbrace{\|\vec{x}_1\|}^{a} \vec{q}_1$

$\vec{x}_2 = b\vec{q}_1 + c\vec{q}_2 = H_1 \vec{x}_2 + H_2 \vec{x}_2 = \vec{q}_1 \underbrace{\vec{q}_1^T \vec{x}_2}_{b} + \vec{q}_2 \underbrace{\vec{q}_2^T \vec{x}_2}_{c}$




$$\begin{array}{ccccccc} \vec{x}_3 = & d & \vec{q}_1 + & e & \vec{q}_2 + & f & \vec{q}_3 \\ & \| & & \| & & \| & \\ & \vec{q}_1 \cdot \vec{x}_3 & & \vec{q}_2 \cdot \vec{x}_3 & & \vec{q}_3 \cdot \vec{x}_3 & \end{array}$$

Sidebar: QR decomposition helps to speedup the OLS estimate computation in the following way:

$\vec{b} = (X^T X)^{-1} X^T \vec{y}$ very expensive operation

$\Downarrow$

$X^T \cancel{X} \vec{b} = X^T \vec{y} \Rightarrow (QR)^T QR\vec{b} = (QR)^T \vec{y} \Rightarrow R^T \overbrace{\cancel{Q^T Q}}^{I} R\vec{b} = R^T \overbrace{Q^T \vec{y}}^{\vec{z}}$

$\Rightarrow R^{T^{-1}} R^T R\vec{b} = R^{T^{-1}} R^T \Rightarrow R\vec{b} = \vec{z}$ by back substitution.

e.g. $p+1=2$

$$\Rightarrow \underbrace{\begin{bmatrix} a & b & d \\ 0 & c & e \\ 0 & 0 & f \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}}_{\vec{b}} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix}$$

$\Rightarrow f b_2 = z_2 \Rightarrow b_2 = z_2/f$

$\Rightarrow c b_1 + e b_2 = z_1 \Rightarrow c b_1 + e \dfrac{z_2}{f} = z_1$

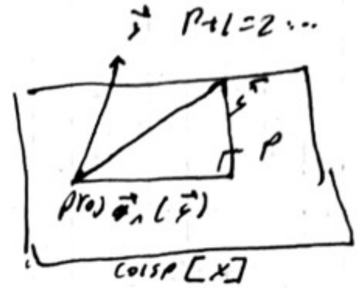
$\Rightarrow b_1 = \dfrac{1}{c}\left(z_1 - e\dfrac{z_2}{f}\right), \dots$ etc

$SST = SSR + SSE \Rightarrow SSR \uparrow \Leftrightarrow SSE \downarrow$

$R^2 \uparrow \Leftrightarrow RMSE \downarrow$

fixed quantity, only a function of the $y$-vector

$X = QR$




$$\hat{\vec{y}} = H\vec{y} = QQ^T\vec{y} = \sum_{j=0}^{p} \text{proj}_{\vec{q}_j}(\vec{y})$$

$$\|\hat{\vec{y}}\|^2 = \sum_{j=0}^{p} \|\text{proj}_{\vec{q}_j}(\vec{y})\|^2 = \|\text{proj}_{\vec{q}_0}(\vec{y})\|^2 + \sum_{j=1}^{p} \|\text{proj}_{\vec{q}_j}(\vec{y})\|^2$$

$$H_0 = \vec{1}(\vec{1}^T\vec{1})^{-1}\vec{1}^T = \frac{1}{n}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

$$= \|\text{proj}_{\vec{1}}(\vec{y})\|^2 = (H_0\vec{y})^T(H_0\vec{y}) = (\bar{y}\vec{1}_n)^T(\bar{y}\vec{1}_n) = \bar{y}^2\vec{1}^T\vec{1} = n\bar{y}^2$$

$$SSR := (\hat{\vec{y}} - \bar{y}\vec{1})^T(\hat{\vec{y}} - \bar{y}\vec{1}) = \hat{\vec{y}}^T\hat{\vec{y}} - \bar{y}\vec{1}^T\hat{\vec{y}} + \bar{y}^2\vec{1}^T\vec{1}$$

$$= \|\hat{\vec{y}}\|^2 - 2\bar{y}\,\hat{\vec{y}}^T\vec{1} + n\bar{y}^2 = \|\hat{\vec{y}}\|^2 - 2n\bar{y}^2 + n\bar{y}^2 = \|\hat{\vec{y}}\|^2 - n\bar{y}^2 = \sum_{j=1}^{p} \|\text{proj}_{\vec{q}_j}(\vec{y})\|^2$$

$$(H\vec{y})^T\vec{1} = \vec{y}^T H\vec{1} = \vec{y}^T\vec{1} = n\bar{y}$$

Pretend your friend gave you a new feature, i.e. a new x-vector $\vec{x}_{new}$ & you want to now update your OLS model to use it

$$X_{new} = [X \mid \vec{x}_{new}]$$

$$SSR_{new} = SSR + \underbrace{\|\text{proj}_{\vec{q}_{new}}(\vec{y})\|^2}_{\geq 0} \Rightarrow SSR_{new} \geq SSR \Leftrightarrow SSE_{new} \leq SSE$$

Now your friend says "btw I made up that vector, it's just a bunch of random nonsense!" Any new column vector in $X$ would have the ostensible effect of improving your model. If that new column vector is independent of the true causal inputs to $y$ (i.e. the $z$'s), we call this "overfitting".

Let's keep going. Your friend keeps supplying you with more and more garbage vectors. What happens when you have the same number of vectors $p+1 = n$

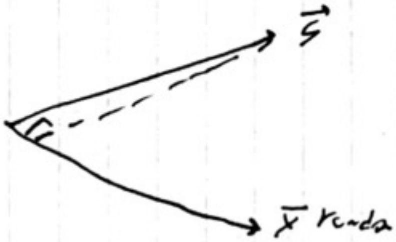
$X_*$ will be $n \times n$ and invertible so $\text{colsp}[X_*] = \mathbb{R}^n$

$$H_* = X_*(X_*^TX_*)^{-1}X_*^T = X_*X_*^{-1}X_*^{T^{-1}}X_*^T = I$$

$$\hat{\vec{y}} = H_*\vec{y} = \vec{y} \Rightarrow \vec{e} = \vec{0}_n \Rightarrow SSE = 0 \Leftrightarrow R^2 = 1 \Leftrightarrow RMSE = 0$$

"perfect fit" or "maximal overfitting"

How did we get into this mess? Consider a random vec $X$ - random




added fake fit (over-fit)
(fake component of SSR)

Overfitting becomes a problem with lots of features relative to n. If you have a small number of features relative to n it's not too bad (i.e. It won't reduce your predictive accuracy)

We proved this in the context of OLS regression, but this is true in every modelling context, overfitting increases "generalization error" which is error on future predictions.

$$\underline{H} = QR^T$$

$$\vec{b} = (X^TX)^{-1}X^T\vec{y} = ((QR)^T(QR))^{-1}(QR)^T\vec{y}$$

$$= (R^TQ^TQR)^{-1}R^TQ^T\vec{y} = (R^TR)^{-1}R^TQ^T\vec{y}$$

$$= R^{-1}R^{T^{-1}}R^TQ^T\vec{y} = R^{-1}Q^T\vec{y}$$

$$\vec{b}a = Q^T\vec{y} = \begin{bmatrix} \vec{q}_0^T\vec{y} \\ q_1^T\vec{y} \\ \vdots \\ \vec{q}_p^T\vec{y} \end{bmatrix}$$