Mathematical Tools
for Neural and Cognitive Science

Fall semester, 2025

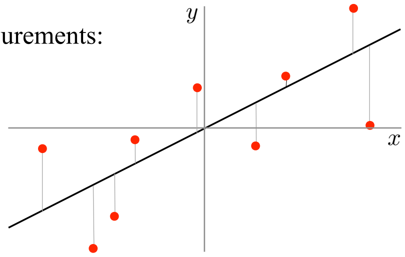
Section 2: Least Squares

Least squares regression:

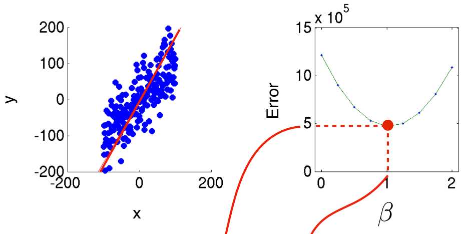
“objective” or “error" function

$$\min_{\beta} \boxed{\sum_n (y_n - \beta x_n)^2}$$

In the space of measurements:



[Gauss, 1795 - age 18!]



$$\hat{\beta} = \arg\min_{\beta} \sum_n (y_n - \beta x_n)^2$$

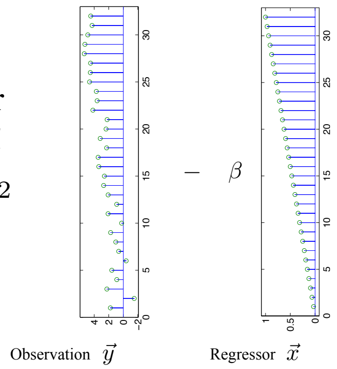
$$\min_{\beta} \sum_n (y_n - \beta x_n)^2$$

$$\min_{\beta} \sum_{n} (y_n - \beta x_n)^2$$

can solve this with calculus... *[on board]*

... or, with linear algebra!

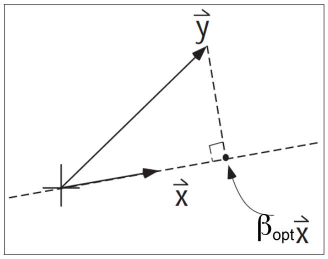
$$\min_{\beta} ||\vec{y} - \beta\vec{x}||^2$$



$$\min_{\beta} ||\vec{y} - \beta\vec{x}||^2$$

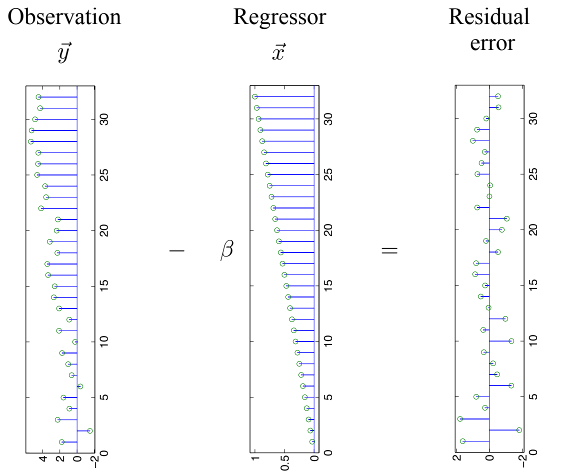
Geometry:

Note: this is a 2-D cartoon of the N-D vectors, not the two-dimensional *(x,y)* measurement space of previous plots!



Note: partition of sum of squared data values:

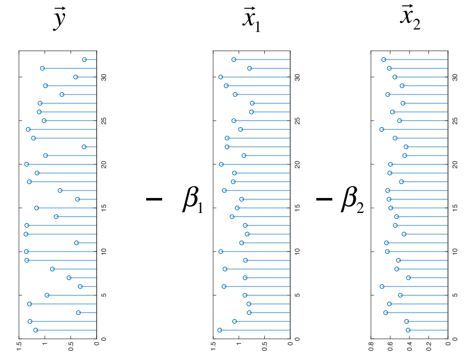
$$||\vec{y}||^2 = \underbrace{\boxed{||\beta_{\text{opt}}\vec{x}||^2}}_{\text{explained}} + \underbrace{\boxed{||\vec{y} - \beta_{\text{opt}}\vec{x}||^2}}_{\text{residual}}$$



Multiple regression:

$$\min_{\vec{\beta}} ||\vec{y} - \sum_k \beta_k \vec{x}_k||^2 = \min_{\vec{\beta}} ||\vec{y} - X\vec{\beta}||^2$$

2D example:

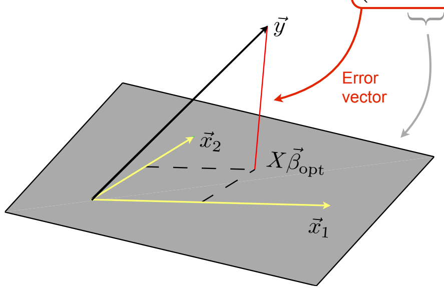


(observation) (regressor 1) (regressor 2)

Solution via the “Orthogonality Principle”:

Construct matrix X, containing columns $\vec{x}_1$ and $\vec{x}_2$

Orthogonality: $X^T \boxed{\left(\vec{y} - X\vec{\beta}\right)} = \vec{0}$



2D vector space containing all linear combinations of $\vec{x}_1$ and $\vec{x}_2$

Alternatively, can solve using SVD...

$$\min_{\vec{\beta}} ||\vec{y} - X\vec{\beta}||^2 = \min_{\vec{\beta}} ||\vec{y} - USV^T\vec{\beta}||^2$$

$$= \min_{\vec{\beta}} ||U^T\vec{y} - SV^T\vec{\beta}||^2$$

$$= \min_{\vec{\beta}^*} ||\vec{y}^* - S\vec{\beta}^*||^2$$

$$\text{where} \quad \vec{y}^* = U^T\vec{y}, \quad \vec{\beta}^* = V^T\vec{\beta}$$

Solution: $\beta^*_{\text{opt},k} = y^*_k / s_k$, for each k

or $\vec{\beta}^*_{\text{opt}} = S^{\#}\vec{y}^* \quad \Rightarrow \vec{\beta}_{\text{opt}} = VS^{\#}U^T\vec{y}$

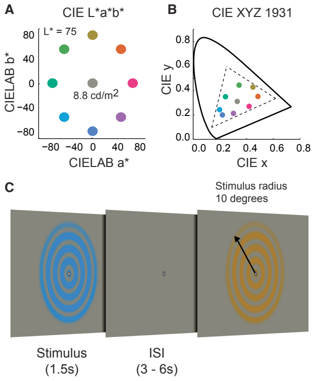
[on board: transformations, elliptical geometry]

Linear regression example: Brouwer & Heeger (2009)

- Examined coding of color in several visual cortical regions
- Used fMRI
- Resulting dataset is 4-d (x/y/z sampled at 3 mm, t sampled at 1.5 s)
- Each data element (“voxel” or volume element) indirectly measures neural activity of 100,000s of neurons via blood oxygenation
- A brief spike in neural activity leads to a BOLD response lasting as much as 12 s (i.e., 8 “TR”s)
- We treat the transformation from neural activity to measured BOLD signal as linear (and the response to a spike as the same at all times: next chapter: this is the BOLD impulse response or HIRF)

Linear regression example: Brouwer & Heeger (2009)

8 “equally spaced” colors:



Linear regression example: Brouwer & Heeger (2009)

Step 1: Estimate the HIRF for each voxel using the MR response from that voxel (a $T \times 1$ vector M), based on a “design matrix” (D, a $T \times 8$ matrix) and the HIRF we hope to estimate (H, an 8×1 vector giving the 12 s response to a brief stimulus): $M \approx DH$, where D looks like:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ & & & & \vdots & & & \end{bmatrix}$$

Solve by linear regression: $\hat{H} = D^{\#}M$, average $\hat{H}$ across an ROI

Linear regression example: Brouwer & Heeger (2009)

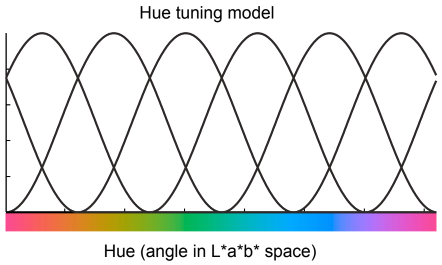
Step 2: Estimate the responses of each voxel to each of the 8 stimulus colors (an 8×1 vector R), based on another "design matrix" (D_2, a $T \times 8$ matrix) and the voxel MR time course (M, a $T \times 1$ vector): $M \approx D_2 R$, where D_2 looks like:

$$D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & h_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_4 & 0 \\ 0 & 0 & h_1 & 0 & 0 & 0 & h_5 & 0 \\ 0 & 0 & h_2 & 0 & 0 & 0 & h_6 & 0 \\ 0 & 0 & h_3 & 0 & 0 & 0 & h_7 & 0 \\ 0 & 0 & h_4 & 0 & 0 & 0 & h_8 & 0 \\ & & & & \vdots & & & \end{bmatrix}$$

Solve by linear regression: $\hat{R} = D_2^{\#} M$, repeat for every voxel

Linear regression example: Brouwer & Heeger (2009)

Step 3: Forward model of voxel tuning. Assume there are 6 "channels" sensitive to different ranges of color:



Each of the 8 stimulus colors results in a vector of 8 channel responses.

Linear regression example: Brouwer & Heeger (2009)

Step 3: Split the data into two subsets (for "train" and "test") B_1 and B_2. These are $m \times n$ matrices, where m is the number of voxels in the ROI and n is the number of response measurements (from Step 2), which is equal to the number of stimulus colors times the number of runs included in that subset of the data.

We seek a weight matrix W, and $m \times 6$ matrix representing the amount by which each channel contributes to each voxel's response (e.g., the proportion of neurons in that voxel belonging to each channels).

From the channel tuning functions, we know how strongly each channel responds to each stimulus, C_1, a $6 \times n$ matrix.

Thus: $B_1 \approx W C_1$ and estimate W by linear regression.

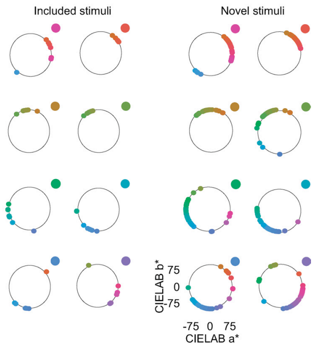
Linear regression example: Brouwer & Heeger (2009)

Step 3, continued: Now consider the "test" dataset B_2. It's still true by the model that $B_2 \approx WC_2$, where C_2 is just like C_1, except that it will represent the series of colors presented in B_2. We have from the "training" dataset an estimate of W.

This time, we treat C_2 as unknown and W as known. We use linear regression to estimate $\hat{C}_2 = W^{\#}B_2$. This estimates the channel responses to every presented stimulus in this subset of the data. That is, it *decodes* what stimulus was on the screen from the neural data. We can then compare the decoded color to the true color presented on each trial.

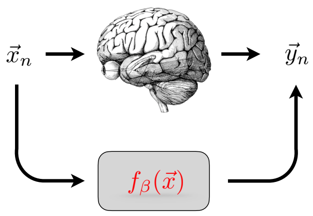
Linear regression example: Brouwer & Heeger (2009)

The decoded color:



Fitting a parametric model (general)

Experimental Data:



Model:

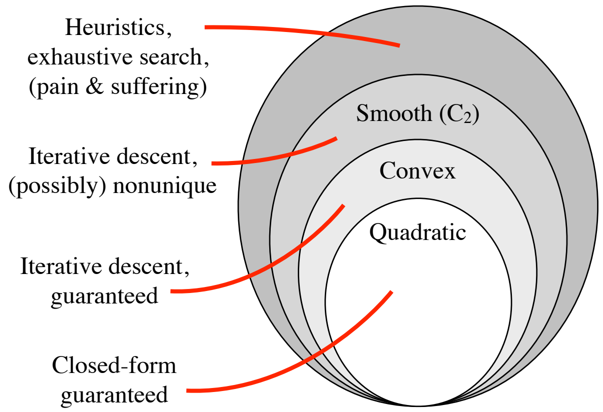
To fit model $f_\beta(\vec{x})$ to data $\{\vec{x}_n, \vec{y}_n\}$,

optimize parameters β to minimize an error function:

$$\min_\beta \sum_n E\left(\vec{y}_n, f_\beta(\vec{x}_n)\right)$$

Ingredients: data, model, error function, optimization method

Optimization



Heuristics, exhaustive search, (pain & suffering)

Iterative descent, (possibly) nonunique

Iterative descent, guaranteed

Closed-form guaranteed

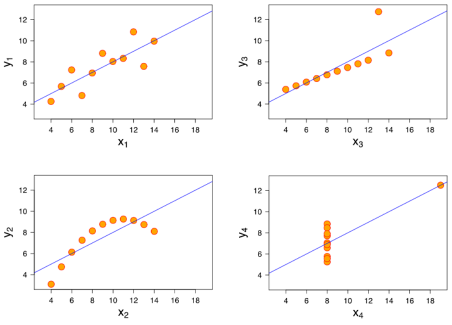
Smooth (C2)

Convex

Quadratic

Interpretation warning: fitting a line does not guarantee data actually lie along a line

These 4 data sets give the same regression fit, and same error:



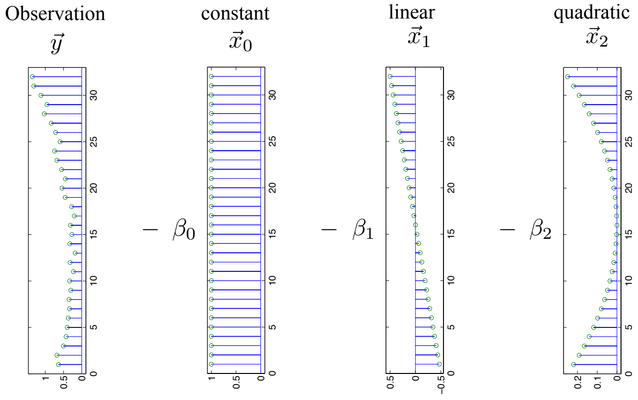
(need a better error metric)

(need a better model)

(need a better experiment)

[Anscombe, 1973]

Polynomial regression



Observation $\vec{y}$

constant $\vec{x}_0$

linear $\vec{x}_1$

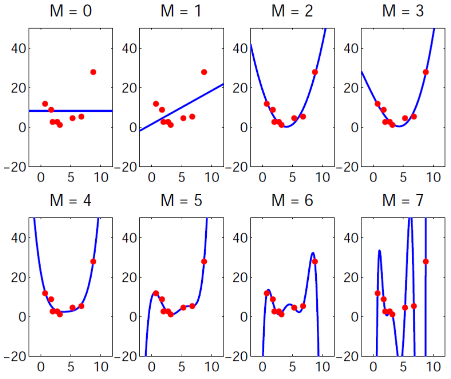
quadratic $\vec{x}_2$

$- \; \beta_0$

$- \; \beta_1$

$- \; \beta_2$

Polynomial regression - how many terms?



(to be continued, when we get to “statistics”...)

Weighted Least Squares

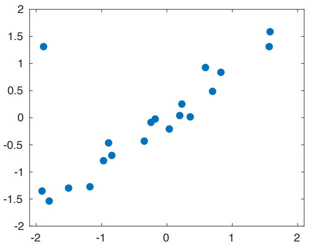
$$\min_{\beta} \sum_{n} [w_n(y_n - \beta x_n)]^2$$

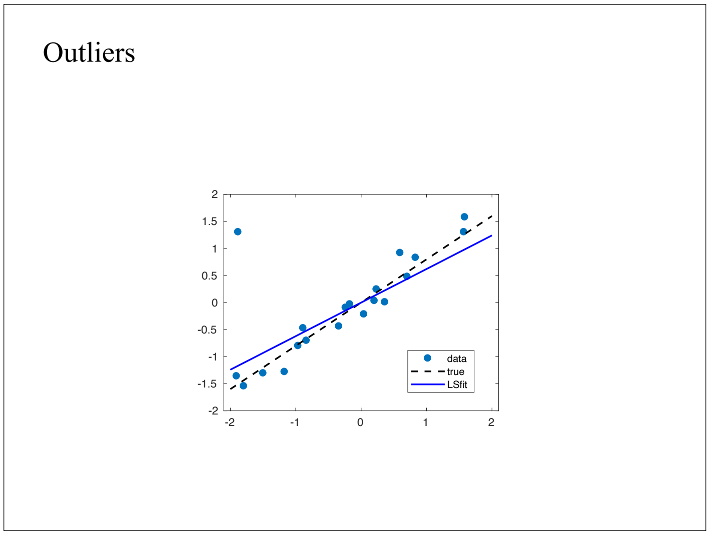
$$= \min_{\beta} ||W(\vec{y} - \beta \vec{x})||^2$$

diagonal matrix

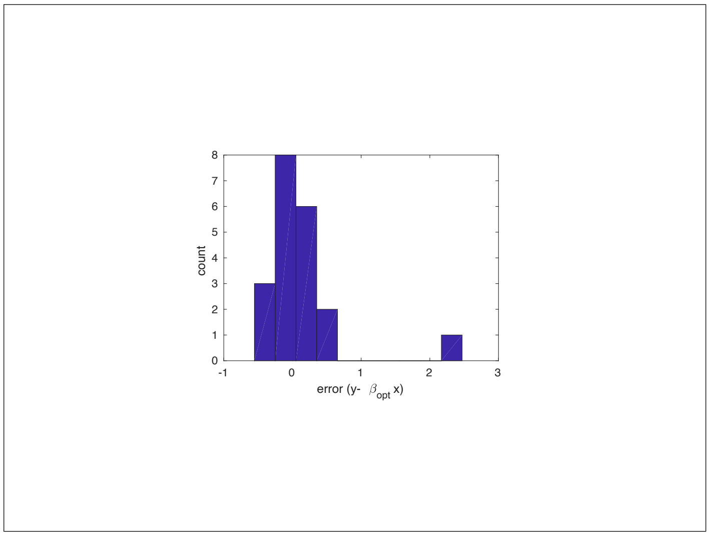
Solution via simple extensions of basic regression solution
(i.e., let $\vec{y}^* = W\vec{y}$ and $\vec{x}^* = W\vec{x}$ then solve for β)

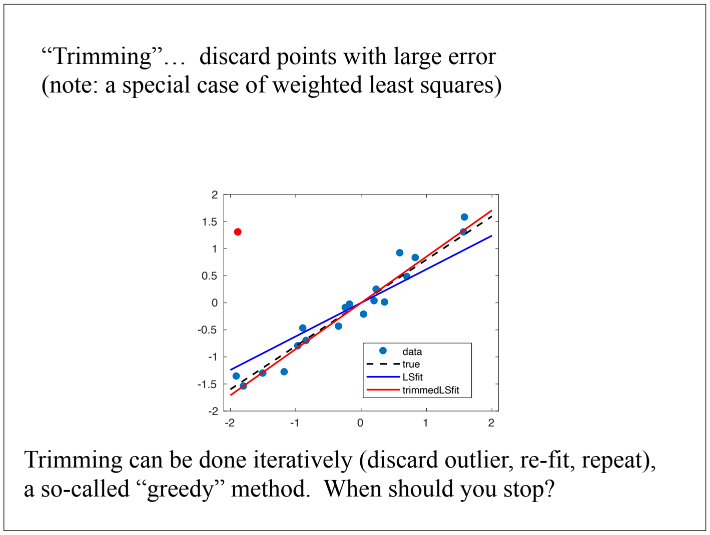
Outliers





Outliers

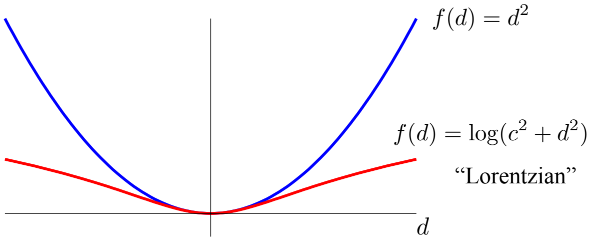




“Trimming”… discard points with large error
(note: a special case of weighted least squares)

Trimming can be done iteratively (discard outlier, re-fit, repeat), a so-called “greedy” method. When should you stop?

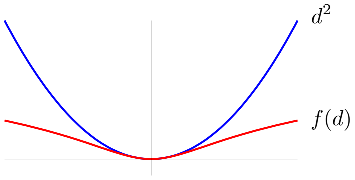
More generally, use a "robust" error metric.
For example:



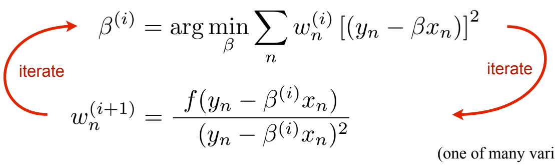
Note: generally can't obtain solution directly (i.e., requires an iterative optimization procedure, such as gradient descent).

In some cases, can use iteratively re-weighted least squares (IRLS)...

Iteratively Re-weighted Least Squares (IRLS)



initialize: $w_n^{(0)} = 1$



$$\beta^{(i)} = \arg\min_{\beta} \sum_n w_n^{(i)} \left[(y_n - \beta x_n)\right]^2$$

$$w_n^{(i+1)} = \frac{f(y_n - \beta^{(i)} x_n)}{(y_n - \beta^{(i)} x_n)^2}$$

(one of many variants)

Constrained Least Squares

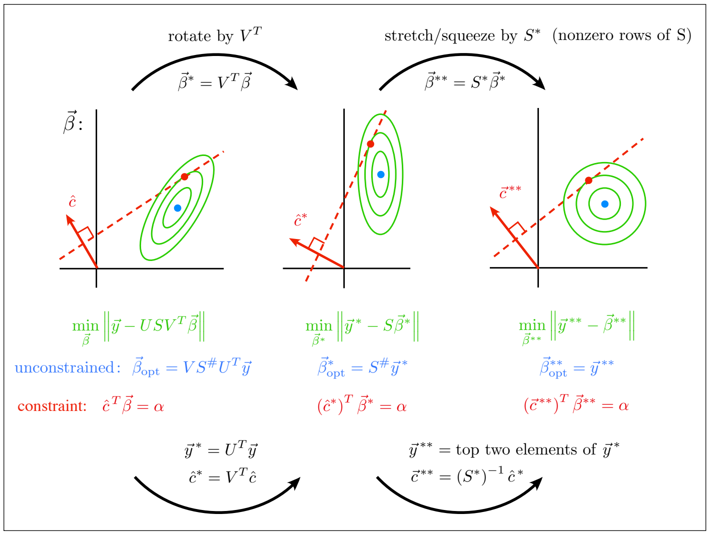
Linear constraint:

$$\arg\min_{\vec{\beta}} \left\| \vec{y} - X\vec{\beta} \right\|^2, \quad \text{where} \quad \vec{c}^T \vec{\beta} = 1$$

Quadratic constraint:

$$\arg\min_{\vec{\beta}} \left\| X\vec{\beta} \right\|^2, \quad \text{where} \quad \left\| \vec{\beta} \right\|^2 = 1$$

Can be solved exactly using linear algebra (SVD)...
[on board, with geometry]


rotate by V^T
$\vec{\beta}^* = V^T\vec{\beta}$
stretch/squeeze by S^* (nonzero rows of S)
$\vec{\beta}^{**} = S^*\vec{\beta}^*$
$\vec{\beta}:$
$\hat{c}$
$\hat{c}^*$
$\vec{c}^{**}$

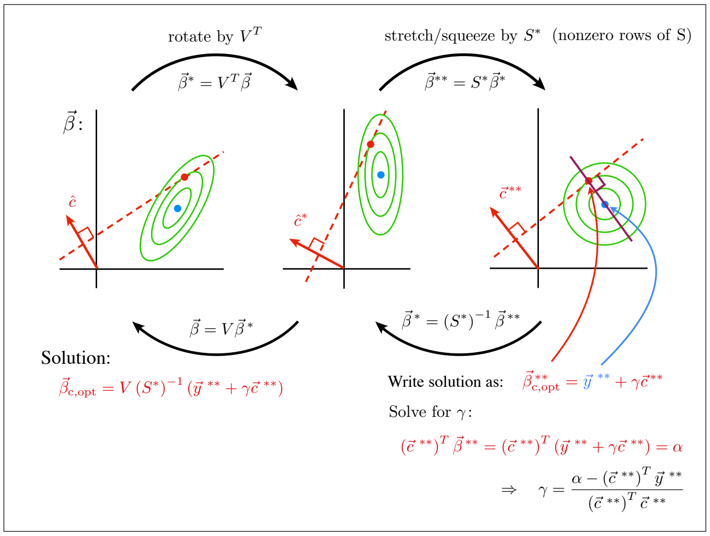
$\min_{\vec{\beta}} \left\| \vec{y} - USV^T\vec{\beta} \right\|$ $\min_{\vec{\beta}^*} \left\| \vec{y}^* - S\vec{\beta}^* \right\|$ $\min_{\vec{\beta}^{**}} \left\| \vec{y}^{**} - \vec{\beta}^{**} \right\|$

unconstrained: $\vec{\beta}_{\text{opt}} = VS^{\#}U^T\vec{y}$ $\vec{\beta}^*_{\text{opt}} = S^{\#}\vec{y}^*$ $\vec{\beta}^{**}_{\text{opt}} = \vec{y}^{**}$

constraint: $\hat{c}^T\vec{\beta} = \alpha$ $(\hat{c}^*)^T\,\vec{\beta}^* = \alpha$ $(\vec{c}^{**})^T\,\vec{\beta}^{**} = \alpha$

$\vec{y}^* = U^T\vec{y}$
$\hat{c}^* = V^T\hat{c}$

$\vec{y}^{**}$ = top two elements of $\vec{y}^*$
$\vec{c}^{**} = (S^*)^{-1}\,\hat{c}^*$


rotate by V^T
$\vec{\beta}^* = V^T\vec{\beta}$
stretch/squeeze by S^* (nonzero rows of S)
$\vec{\beta}^{**} = S^*\vec{\beta}^*$
$\vec{\beta}:$
$\hat{c}$
$\hat{c}^*$
$\vec{c}^{**}$
$\vec{\beta} = V\vec{\beta}^*$
$\vec{\beta}^* = (S^*)^{-1}\,\vec{\beta}^{**}$

Solution:

$\vec{\beta}_{\text{c,opt}} = V\,(S^*)^{-1}\,(\vec{y}^{\,**} + \gamma\vec{c}^{\,**})$

Write solution as: $\vec{\beta}^{**}_{\text{c,opt}} = \vec{y}^{\,**} + \gamma\vec{c}^{**}$

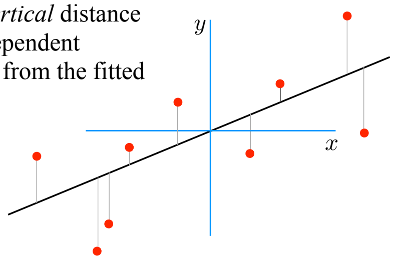
Solve for γ:

$$(\vec{c}^{\,**})^T\,\vec{\beta}^{**} = (\vec{c}^{\,**})^T\,(\vec{y}^{\,**} + \gamma\vec{c}^{\,**}) = \alpha$$

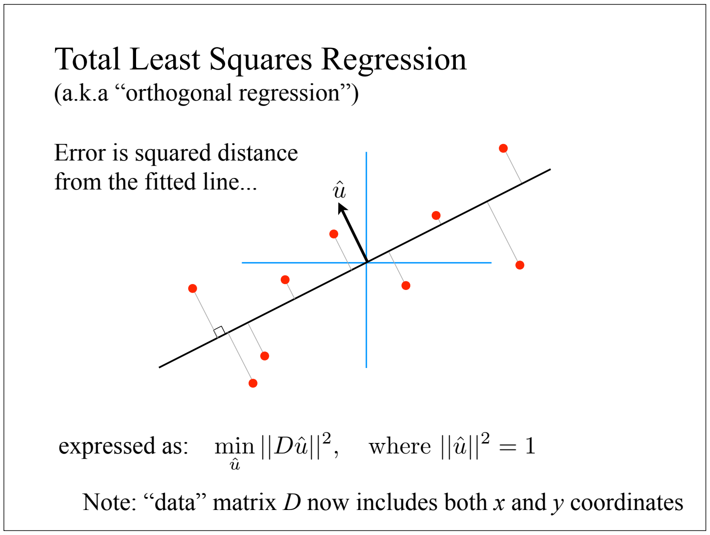
$$\Rightarrow \quad \gamma = \frac{\alpha - (\vec{c}^{\,**})^T\,\vec{y}^{\,**}}{(\vec{c}^{\,**})^T\,\vec{c}^{\,**}}$$

Standard Least Squares regression

Error is *vertical* distance (in the “dependent variable”) from the fitted line...



$$\arg\min_{\beta} ||\vec{y} - \beta\vec{x}||^2$$



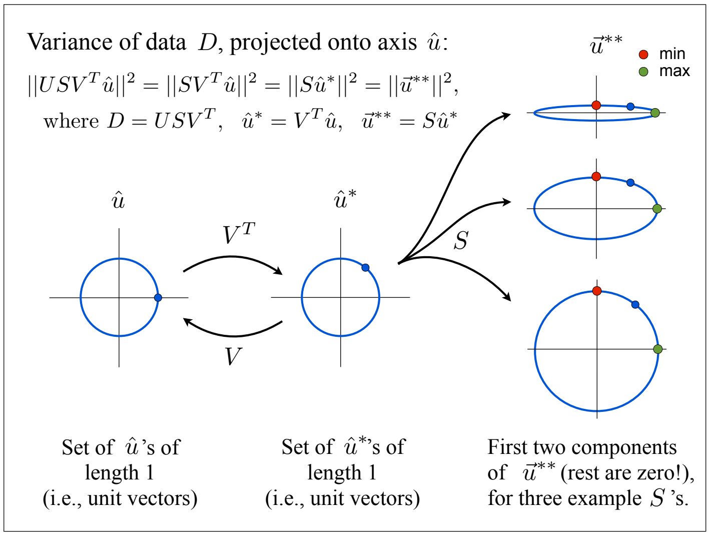
Total Least Squares Regression

(a.k.a “orthogonal regression”)

Error is squared distance from the fitted line...

expressed as: $\min_{\hat{u}} ||D\hat{u}||^2, \quad \text{where } ||\hat{u}||^2 = 1$

Note: “data” matrix D now includes both x and y coordinates



Variance of data D, projected onto axis $\hat{u}$:

$$||USV^T\hat{u}||^2 = ||SV^T\hat{u}||^2 = ||S\hat{u}^*||^2 = ||\vec{u}^{**}||^2,$$

$$\text{where } D = USV^T, \quad \hat{u}^* = V^T\hat{u}, \quad \vec{u}^{**} = S\hat{u}^*$$

Set of $\hat{u}$’s of length 1 (i.e., unit vectors)

Set of $\hat{u}^*$’s of length 1 (i.e., unit vectors)

First two components of $\vec{u}^{**}$ (rest are zero!), for three example S ’s.