Regression and Linear Models

First – a review of known material

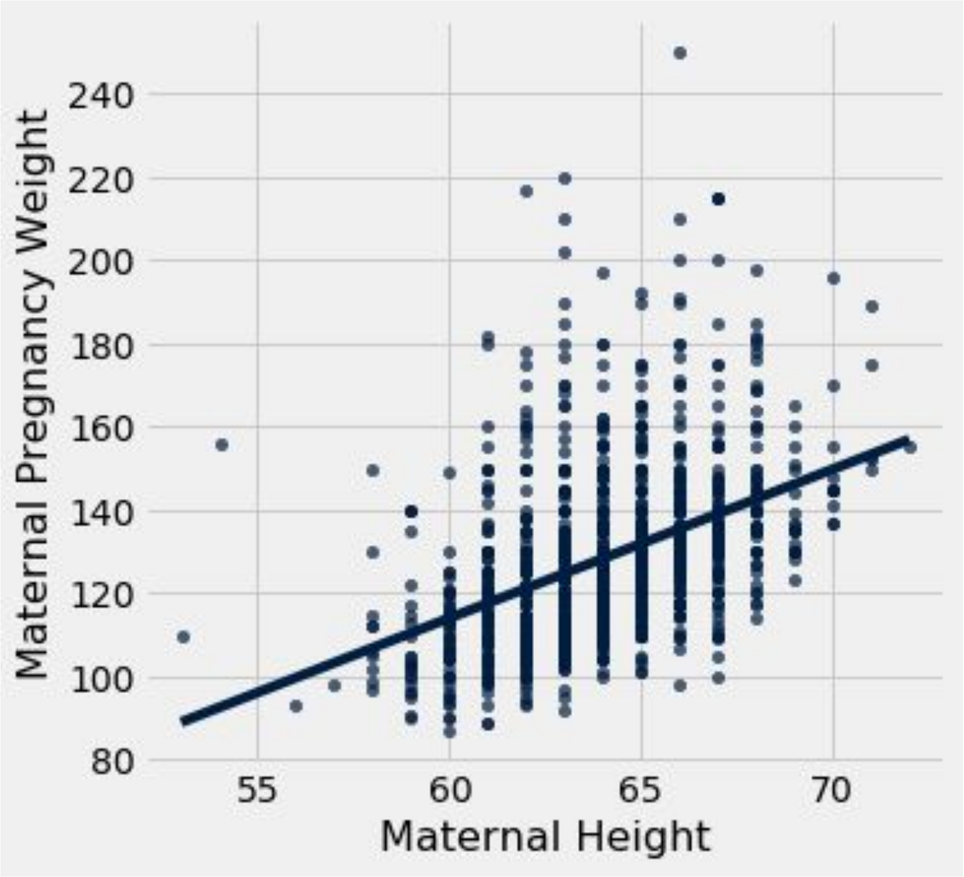
Slides: Data 100, Joey Gonzalez

The Regression Line (Data 8)

In Data 8, you talked about generating the regression line.

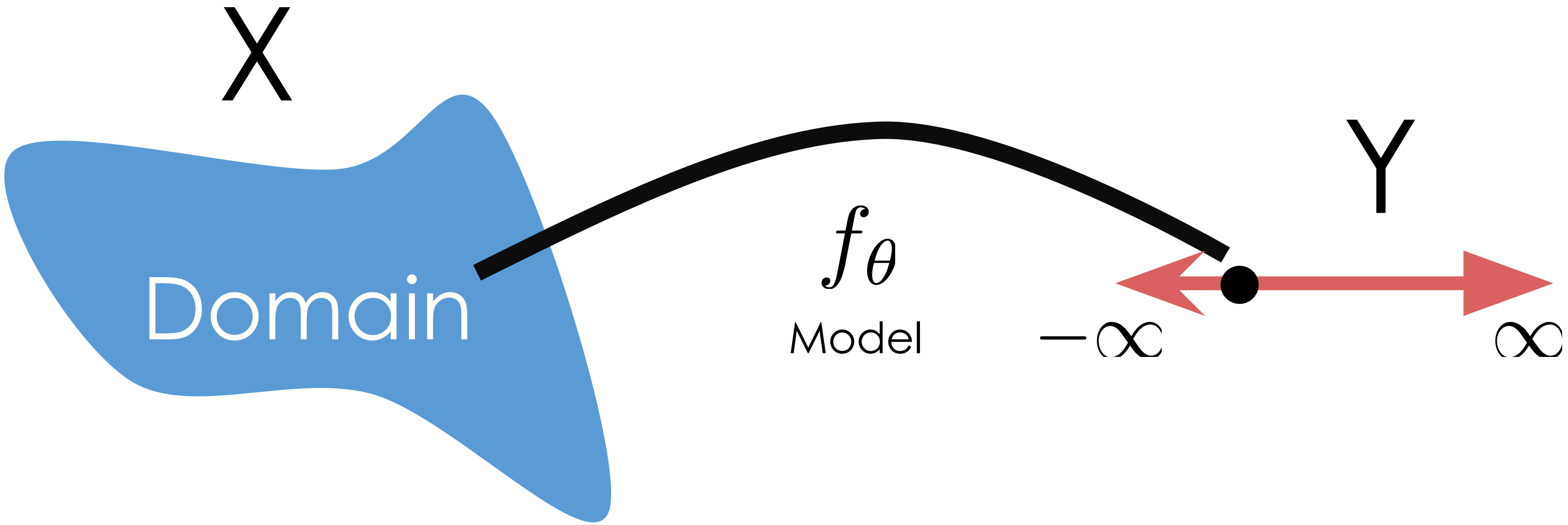
Given scalar data y and x, find m and b that minimizes the mean squared error (a.k.a. L2 Loss).

$$Loss = \left(y - (mx + b)\right)^2$$



Regression

- Estimating relationship between X and Y.
  - Y is a quantitative value.
  - X can be almost anything …



Least Squares Linear Regression

One of the most widely used tools in machine learning and data science

Linear Model

Linear in the Parameters

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

Feature Functions

Squared Loss

Loss Minimization

$$\hat{\theta} = \arg\min \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{d} \theta_j \phi_j(x_i) \right)^2$$

We will return to solving this soon!

Linear Models and Feature Functions

Linear in the Parameters

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

Feature Functions

For Example:

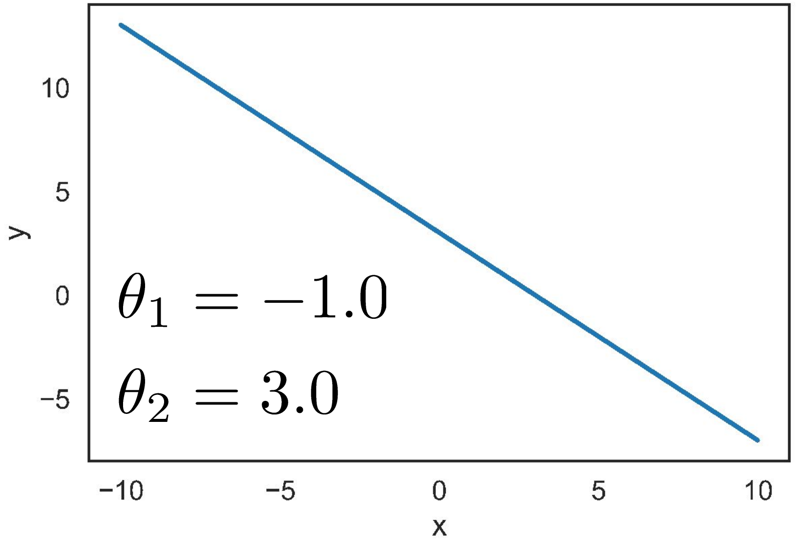
Domain: $x \in \mathbb{R}$

Model: $f_\theta(x) = \theta_1 x + \theta_2$

<u>Features</u>:

$$\phi_1(x) = x$$

$$\phi_2(x) = 1$$



Adding a **“constant” feature function** $\phi_2(x) = 1$ is a common method to introduce an **offset** (also sometimes called **bias**) term.

Linear Models and Feature Functions

Linear in the Parameters

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

Feature Functions

For Example:

Domain: $x \in \{False, True\}^2 \times \mathcal{R}$

Model: $y_i = \theta_1^* + \theta_2^* \mathbb{I}(x_i \; is \; 'Male') + \theta_3^* \mathbb{I}(x_i \; is \; 'Smoker') + \theta_4^* size(x_i)$

Features:

$$\phi_1(x) = \mathbb{I}(x \; is \; 'Male')$$

$$\phi_2(x) = \mathbb{I}(x \; is \; 'Smoker')$$

$$\phi_3(x) = size(x)$$

Indicator functions

$$\phi_1(x) = \mathbb{I}(x \; is \; 'Male')$$

are a common method to transform qualitative data into quantitative data.

Linear Models and Feature Functions

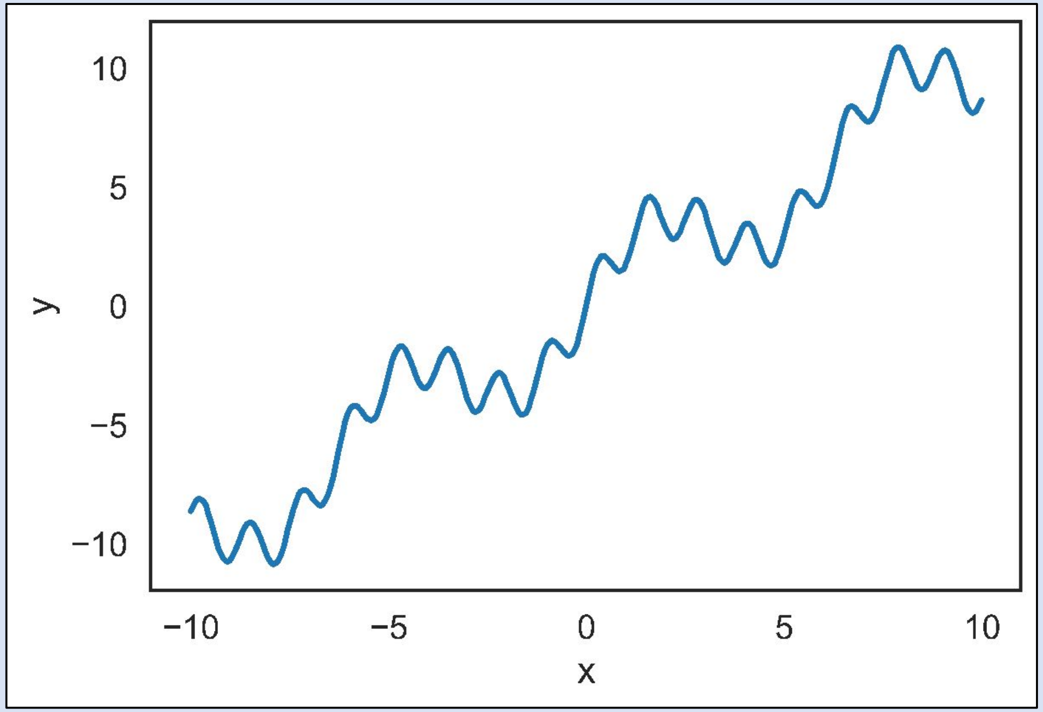
$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

Linear in the Parameters

Feature Functions

Question: Can a linear model do a good job of fitting y and x (to the right)?

A. Yes
B. No
C. Not sure



Linear Models and Feature Functions

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

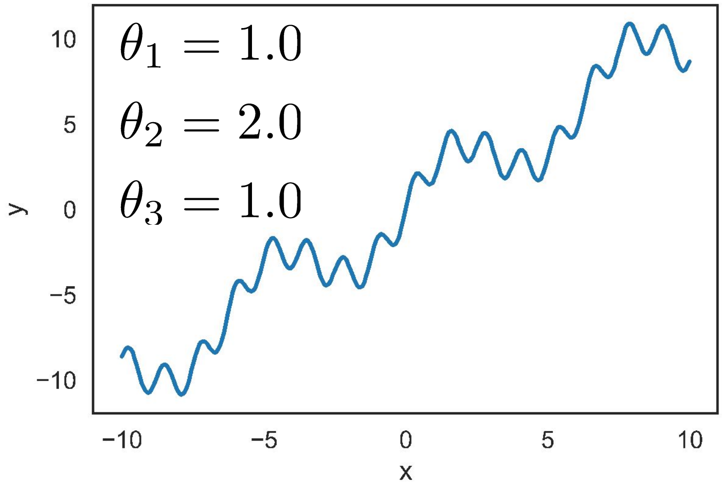
Linear in the Parameters

Feature Functions

Yes: $x \in \mathbb{R}$ $\quad f_\theta(x) = \theta_1 x + \theta_2 \sin(x) + \theta_3 \sin(5x)$

Features:

$$\phi_1(x) = x$$
$$\phi_2(x) = \sin(x)$$
$$\phi_3(x) = \sin(5x)$$



- This is a linear model!

Linear in the parameters

Linear Models and Feature Functions

Linear in the Parameters

Feature Functions

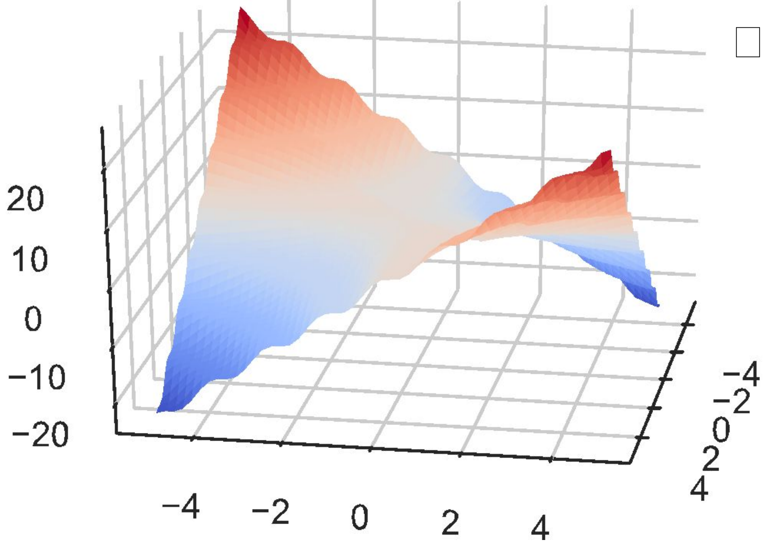
$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

For Example: $x \in \mathbb{R}^2$

$$f_\theta(x) = \theta_1 x_1 x_2 + \theta_2 \cos(x_2 x_1) + \theta_3 \mathbb{I}\left[x_1 > x_2\right]$$

Features:

$$\phi_1(x) = x_1 x_2$$
$$\phi_2(x) = \cos(x_2 x_1)$$
$$\phi_3(x) = \mathbb{I}\left[x_1 > x_2\right]$$



- This is a linear model!

Linear in the parameters

Linear Models and Feature Functions

Linear in the Parameters

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

Feature Functions

Designing feature functions is a big part of machine learning and data science.

Feature Functions

- capture domain knowledge
- contribute to expressivity (and complexity)

Loss Minimization for Linear Models

Linear Models in Matrix Notation

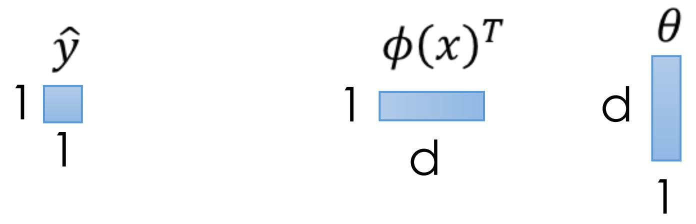
We discussed how our model takes an observation and produces a prediction.

$$\hat{y} = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x)$$

We can also express this in matrix notation:

- $\phi(x)$ is a vector of d features.
- θ is a vector of d parameters.

$$\hat{y} = f_\theta(x) = \phi(x)^T \theta$$



Linear Models in Matrix Notation

Often we'll make predictions over entire datasets: For each x_i, we'll predict y_i using our model.

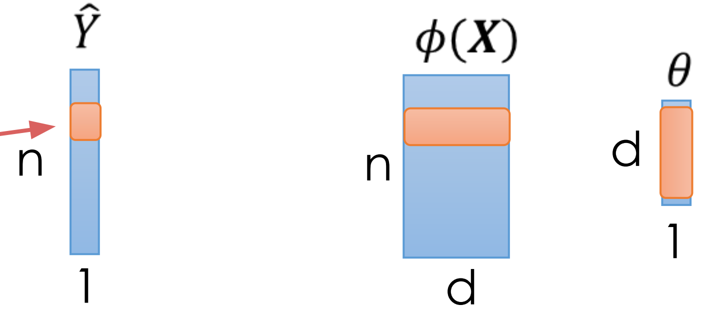
$$\hat{y}_i = f_\theta(x) = \sum_{j=1}^{d} \theta_j \phi_j(x_i)$$

We can also express this in matrix notation:

- $\phi(X)$ is an n x d matrix of features.
- θ is a vector of d parameters.
- $\hat{Y}$ is a vector of n predictions.

$$\hat{Y} = f_\theta(x) = \phi(\boldsymbol{X})\theta$$

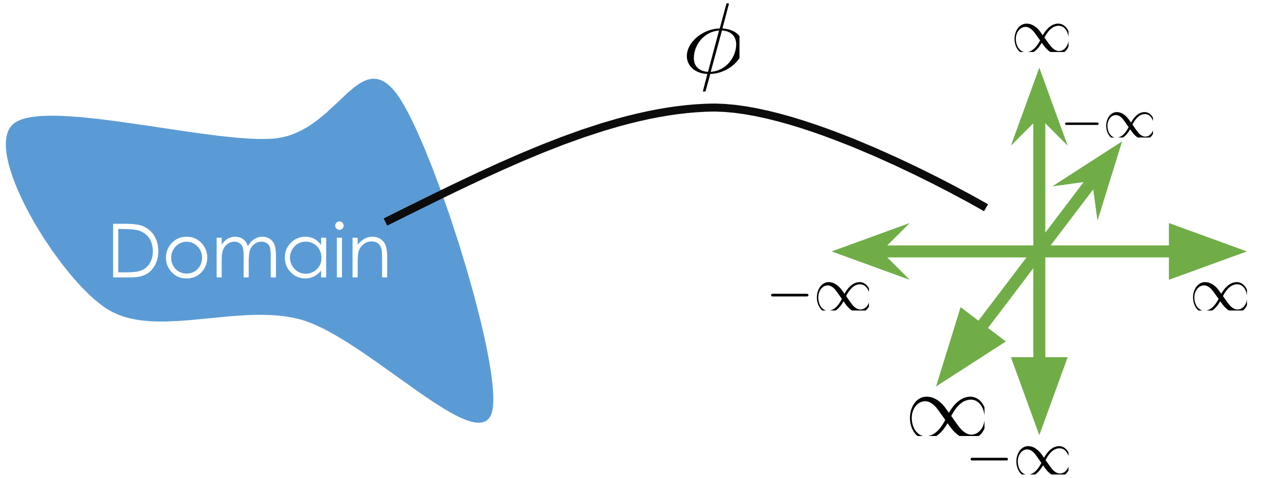
Our prediction for record #i is a linear combination of all d features of record #i.



For notational convenience, we'll often replace $\phi(X)$ by the "feature matrix" $\boldsymbol{\Phi}$.

$$\hat{Y} = \Phi\theta$$

The Feature Matrix Φ



X DataFrame (n x c)

uid	age	state	hasBought	review
0	32	NY	True	"Meh."
42	50	WA	True	"Worked out of the box …"
57	16	CA	NULL	"Hella tots lit…"

$\xrightarrow{\phi}$

$\Phi \in \mathbb{R}^{n \times d}$

AK	…	NY	…	WY	age	hasBought	hasBought missing
0	…	1	…	0	32	1	0
0	…	0	…	0	50	1	0
0	…	0	…	0	16	0	1

Entirely **Quantitative** Values

The Feature Matrix Φ

AK	...	NY	...	WY	age	hasBought	hasBought missing
0	...	1	...	0	32	1	0
0	...	0	...	0	50	1	0
0	...	0	...	0	16	0	1

Entirely **Quantitative** Values

$$\Phi \in \mathbb{R}^{n \times d} = \phi\left[X \right] = \begin{array}{c} \\ n \end{array} \underbrace{\begin{bmatrix} \text{——} \phi(X_{1,\bullet}) \text{——} \\ \text{——} \phi(X_{2,\bullet}) \text{——} \\ \dots \\ \text{——} \phi(X_{n,\bullet}) \text{——} \end{bmatrix}}_{d}$$

DataFrame

Rows of the $\mathbf{\Phi}$ matrix correspond to records (observations).

Columns of the $\mathbf{\Phi}$ matrix correspond to features.

Notation Guide

$A_{i,\bullet}$: row i of matrix A.

$A_{\bullet,j}$: column j of matrix A.

Making Predictions

$$\Phi \in \mathbb{R}^{n \times d} = \phi\big[\,X\,\big] = \begin{array}{c} \\ n \end{array}\underbrace{\begin{pmatrix} \text{---}\, \phi\left(X_{1,\bullet}\right) \,\text{---} \\ \text{---}\, \phi\left(X_{2,\bullet}\right) \,\text{---} \\ \cdots \\ \text{---}\, \phi\left(X_{n,\bullet}\right) \,\text{---} \end{pmatrix}}_{d}$$

DataFrame

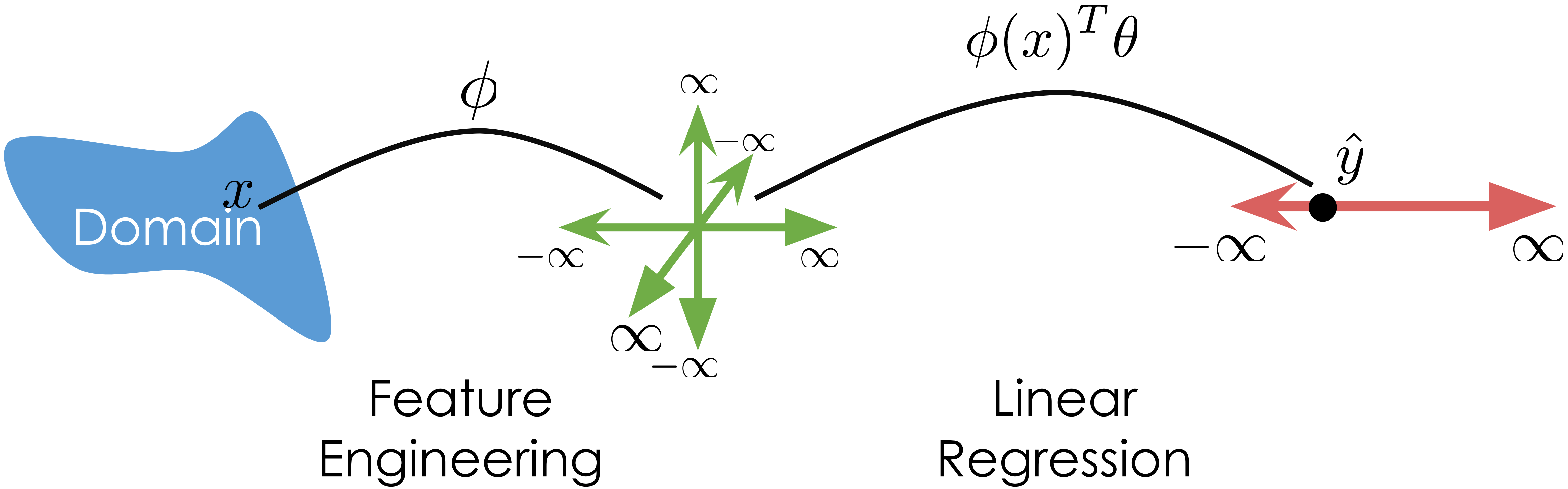
Rows of the $\mathbf{\Phi}$ matrix correspond to records.

Columns of the $\mathbf{\Phi}$ matrix correspond to features.

Prediction

$$\hat{Y} = f_{\hat{\theta}}(X) = \Phi\hat{\theta} = \begin{array}{c} \\ n \end{array}\underbrace{\begin{pmatrix} \text{---}\, \phi\left(X_{1,\bullet}\right) \,\text{---} \\ \boxed{\text{---}\, \phi\left(X_{2,\bullet}\right) \,\text{---}} \\ \cdots \\ \text{---}\, \phi\left(X_{n,\bullet}\right) \,\text{---} \end{pmatrix}}_{d} \boxed{\begin{matrix} | \\ \hat{\theta} \\ | \end{matrix}} = \begin{pmatrix} \hat{y}_1 \\ \boxed{\hat{y}_2} \\ \cdots \\ \hat{y}_n \end{pmatrix}$$

Summary of Notation



Loss Functions

- **Loss function**: a function that characterizes the cost, error, or loss resulting from a particular choice of model or model parameters.
- *Many definitions* of loss functions and the choice of loss function affects the **accuracy** and **computational cost of estimation**.
- The choice of loss function **depends on the estimation task**
  - quantitative (e.g., tip) or qualitative variable (e.g., political affiliation)
  - Do we care about the outliers?
  - Are all errors equally costly? (e.g., false negative on cancer test)

Squared Loss

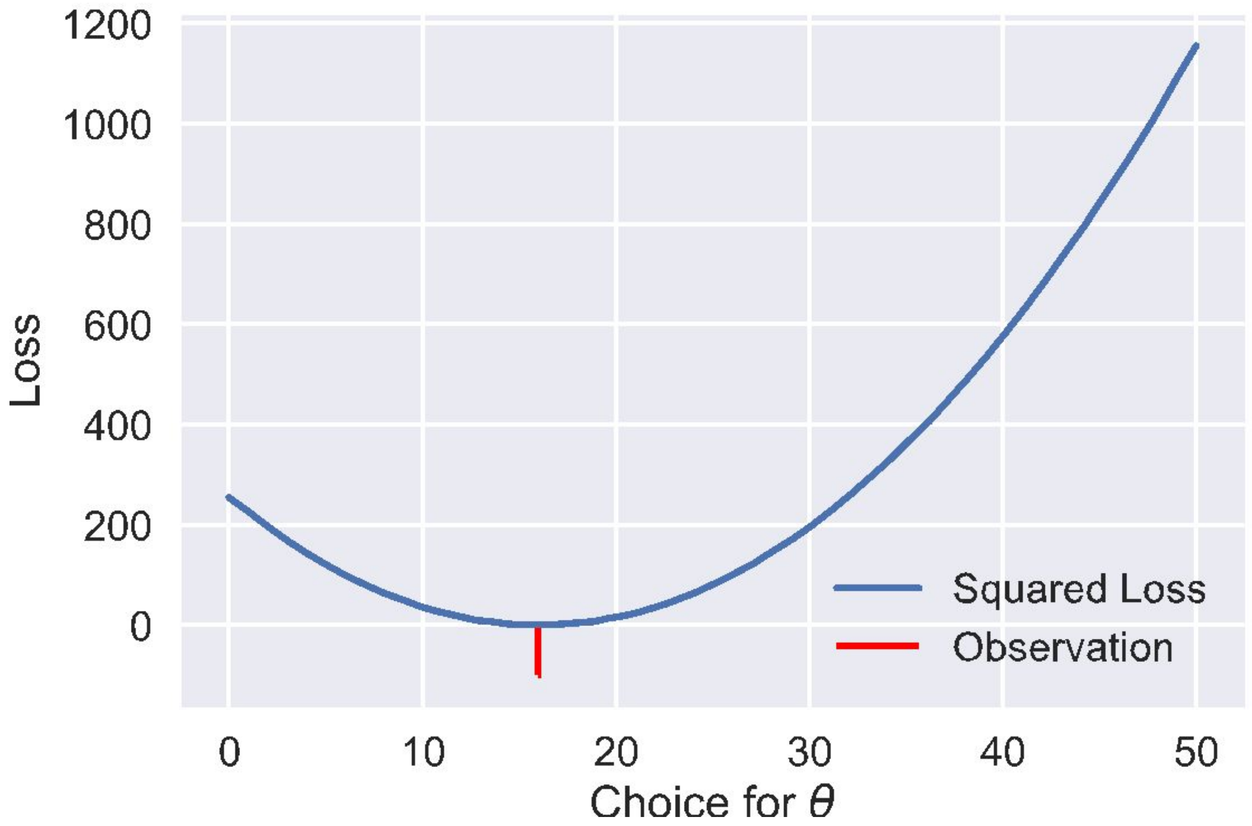
Widely used loss!

The predicted value

The "error" in our prediction

$$L(\theta, y) = (y - \theta)^2$$

An observed data point



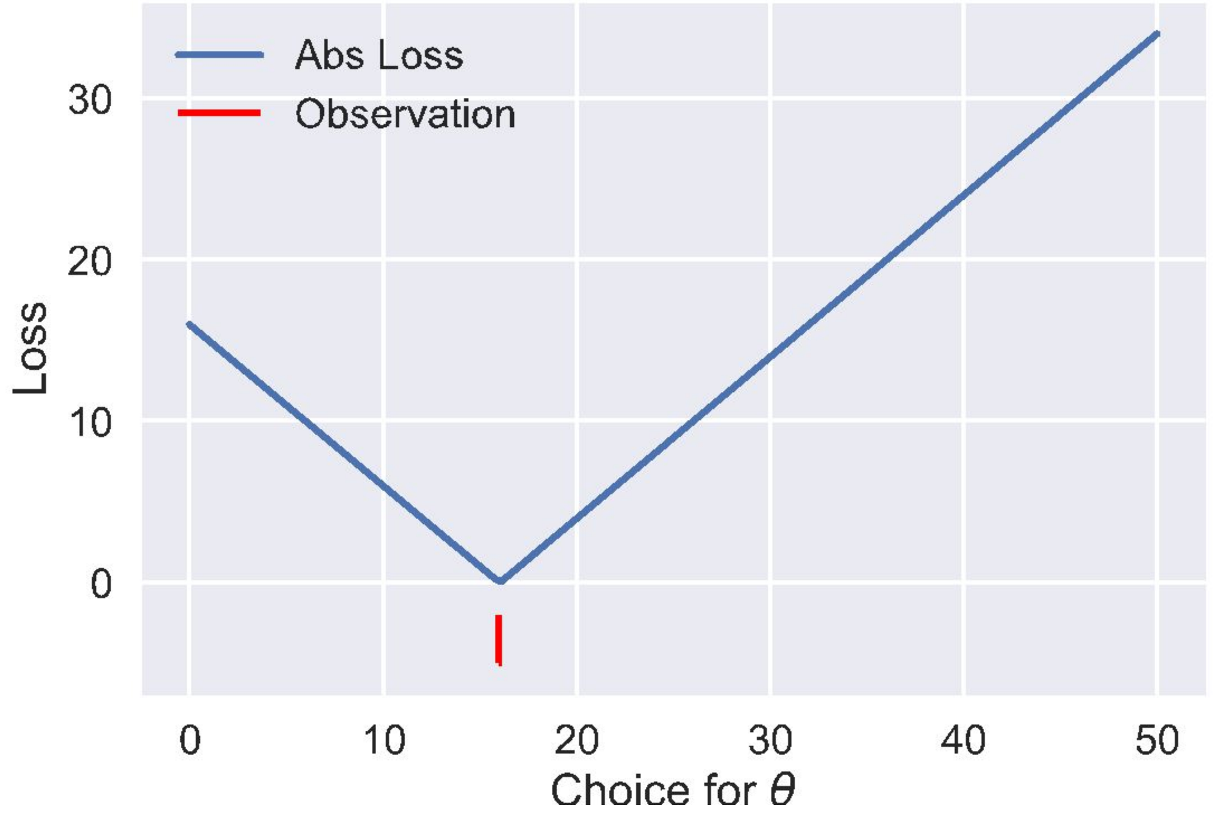
- Also known as the the L^2 loss (pronounced "el two")
- Reasonable?
  - *$\theta = y$ → good prediction → good fit → no loss!*
  - *θ far from y → bad prediction → bad fit → lots of loss!*

Absolute Loss

It sounds worse than it is …

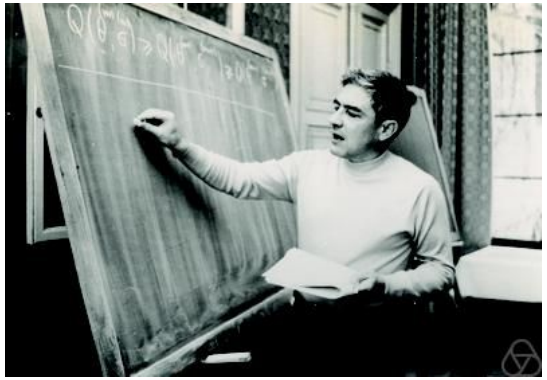
$$L(\theta, y) = |y - \theta|$$

Absolute value



- Also known as the the L^1 loss (pronounced "el one")
- Reasonable?
    - $\theta = y$ good prediction good fit no loss!
    - θ far from y bad prediction bad fit some loss

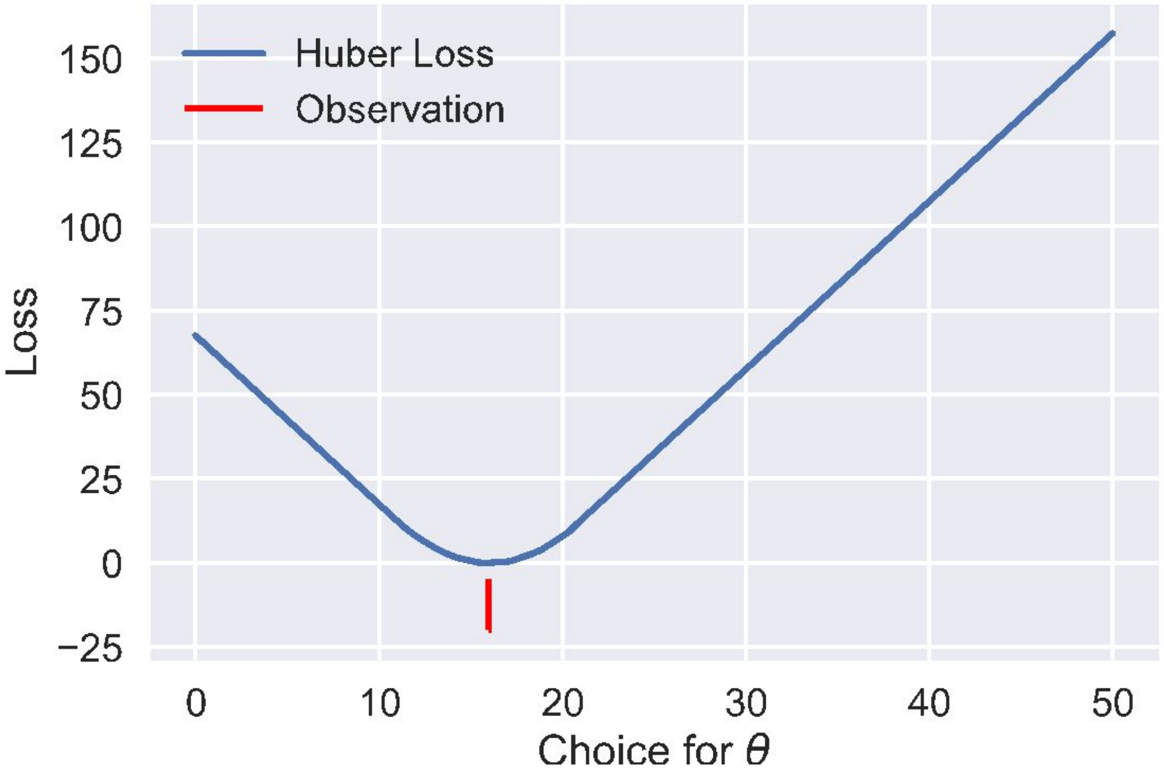
Can you think of another Loss Function?



$$L_\alpha(\theta, y) = \begin{cases} \frac{1}{2}(y-\theta)^2 & |y-\theta| < \alpha \\ \alpha\left(|y-\theta| - \frac{\alpha}{2}\right) & \text{otherwise} \end{cases}$$

Huber Loss

- Parameter α that we need to choose.
- Reasonable?
  - *θ = y → good prediction → good fit → no loss!*
  - *θ far from y → bad prediction → bad fit → some loss*
- A hybrid of the L2 and L1 losses…



The Loss Function in Matrix Notation

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overbrace{\sum_{j=1}^{d}\theta_j\phi_j(x_i)}^{\hat{Y}}\right)^2 = \frac{1}{n}(Y-\hat{Y})^T(Y-\hat{Y})$$

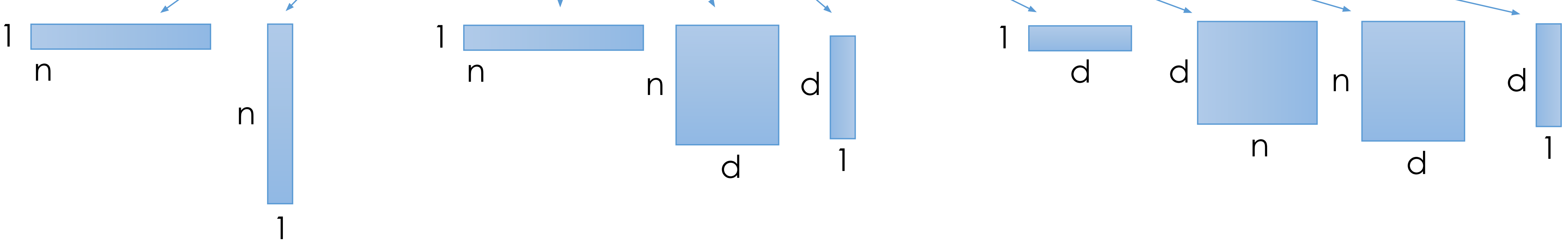
$$\begin{bmatrix} y_1-\hat{y}_1 & y_2-\hat{y}_2 & \cdots & y_n-\hat{y}_n \end{bmatrix}\begin{bmatrix} y_1-\hat{y}_1 \\ y_2-\hat{y}_2 \\ \cdots \\ y_n-\hat{y}_n \end{bmatrix}$$

$$= \frac{1}{n}(Y-\Phi\theta)^T(Y-\Phi\theta)$$

$$= \frac{1}{n}\left(Y^TY - 2Y^T\Phi\theta + \theta^T\Phi^T\Phi\theta\right)$$

The Loss Function in Matrix Notation

$$L(\theta) = \frac{1}{n}\left(Y^TY - 2Y^T\Phi\theta + \theta^T\Phi^T\Phi\theta\right)$$



To minimize, we need to compute the gradient and set it equal to zero.

Minimizing the Loss

$$L(\theta) \ = \frac{1}{n}\left(Y^T Y - 2Y^T \Phi\theta + \theta^T \Phi^T \Phi\theta\right)$$

$$\nabla_\theta \mathrm{L}(\theta) = \left[\frac{\partial L(\theta)}{\partial \theta_1}, \frac{\partial L(\theta)}{\partial \theta_2}, \dots, \frac{\partial L(\theta)}{\partial \theta_d}\right] = [0, 0, \dots, 0]$$

Could expand out $L(\theta)$ and do calculus + algebra, but this would be incredibly tedious! (might be worth doing to test your understanding).

$$L(\theta) = \frac{1}{n}\Big((y_1^2 + y_2^2 + \cdots y_n^2) - 2\big((y_1\phi_{11} + y_2\phi_{21} + \cdots y_n\phi_{n1})\theta_1 + \cdots\big) + \cdots\Big)$$

Instead, let's do everything natively in matrix notation.

Loss minimization

- Review these slides on your own (you can watch former D100 lectures if useful).

Some Useful Matrix Calculus Rules

Let's discuss a couple of rules, that are useful to us.

First: $\nabla_\theta(A\theta) = A^T$, where A and θ are 1 x d and d x 1, respectively.

Transpose because we want gradient to be a column vector!

$$\nabla_\theta(A\theta) = \left[\frac{\partial(A\theta)}{\partial\theta_1}, \frac{\partial(A\theta)}{\partial\theta_2}, \dots, \frac{\partial(A\theta)}{\partial\theta_n}\right]^T$$

$$= \left[\frac{\partial(a_1\theta_1 + a_2\theta_2 + \cdots + a_d\theta_d)}{\partial\theta_1}, \dots \frac{\partial(a_1\theta_1 + a_2\theta_2 + \cdots + a_d\theta_d)}{\partial\theta_d}\right]^T$$

$$= [a_1, a_2, \dots, a_n]^T \quad = A^T$$

Some Useful Matrix Calculus Rules

Second: $\nabla_\theta(\theta^T A\theta) = A\theta + A^T\theta$, where A and θ are d x d and d x 1.

Proof is not hard, but a bit tedious. Not shown here. Similar to first proof.

Useful Matrix Derivative Rules:

(1) $$\nabla_\theta\left(A\theta\right) = A^T$$

Optimizing the Loss Algebraically

Deriving the Normal Equation

$$L(\theta) = \frac{1}{n}\left(Y^T Y - 2Y^T \Phi\theta + \theta^T \Phi^T \Phi\theta\right)$$

Rule 1 Rule 2

Taking the Gradient of the loss

$$\nabla_\theta L(\theta) = -\frac{2}{n}\Phi^T Y + \frac{2}{n}\Phi^T \Phi\theta$$

Useful Matrix Derivative Rules:

(1) $\nabla_\theta \left(A\theta\right) = A^T$

(2) $\nabla_\theta \left(\theta^T A\theta\right) = A\theta + A^T\theta$

Setting the gradient equal to 0 and solving for θ:

$$0 = -\frac{2}{n}\Phi^T Y + \frac{2}{n}\Phi^T \Phi\theta \longrightarrow$$

$$\hat{\theta} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y$$

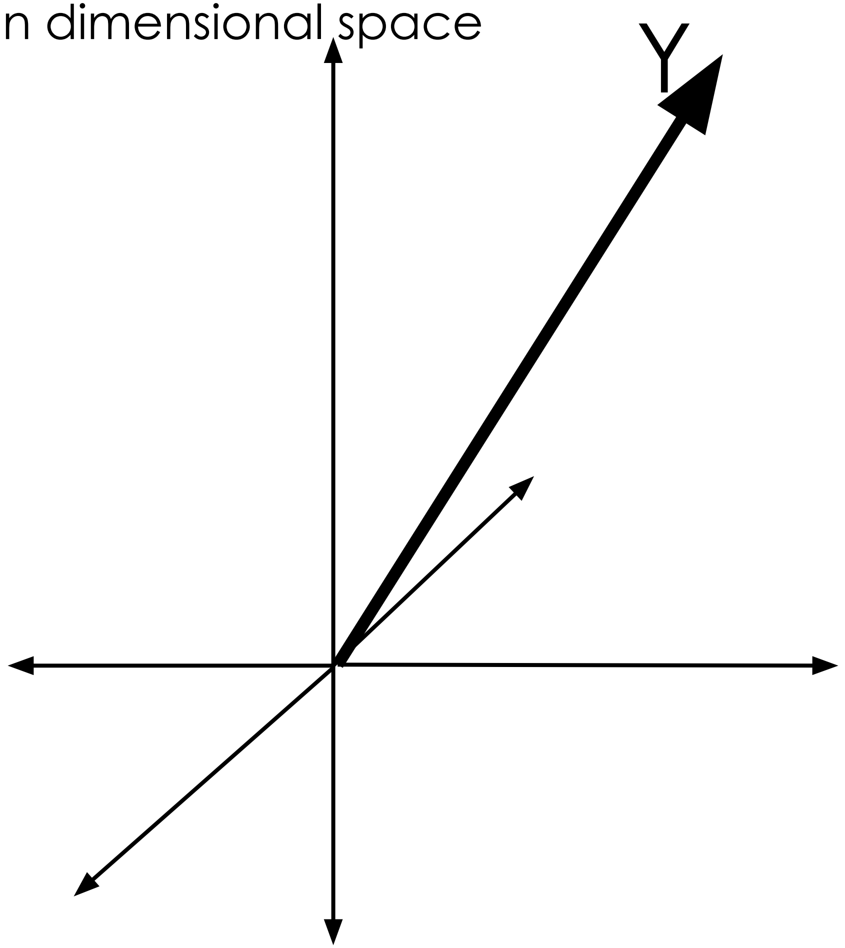
"Normal Equation"

Optimizing the Loss Geometrically

- There is an alternate derivation for the normal equation.
- This one provides much more intuition, but requires a deeper understanding of linear algebra.
- Understanding this is required for the course. Will vary widely in how much effort it takes to fully grok.

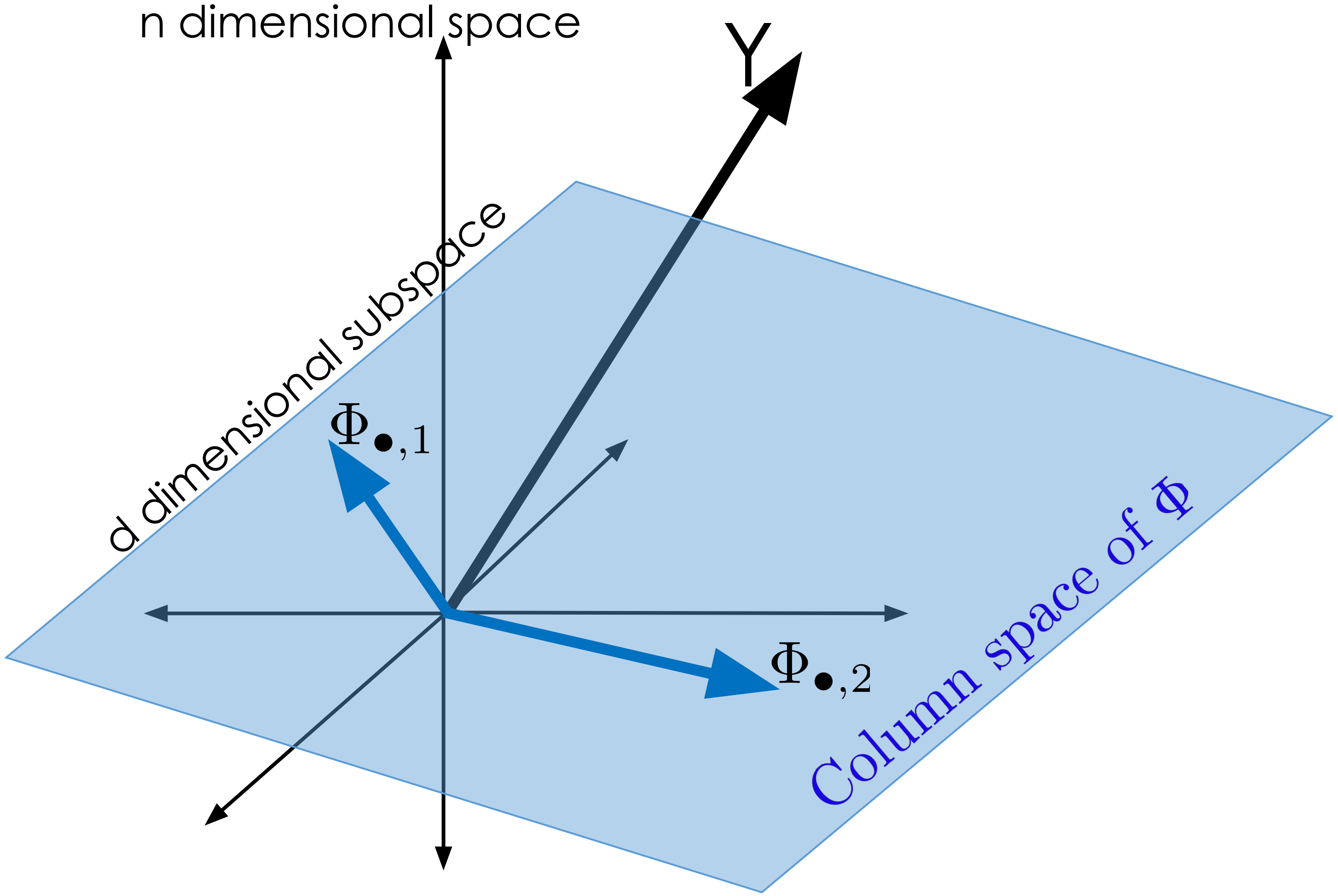
Optimizing the Loss Geometrically

- Our observations Y form a single vector in an n-dimensional space.
- Maybe it's the observed weight of all n people in Berkeley: [120, 190, 210, 9.3, …]



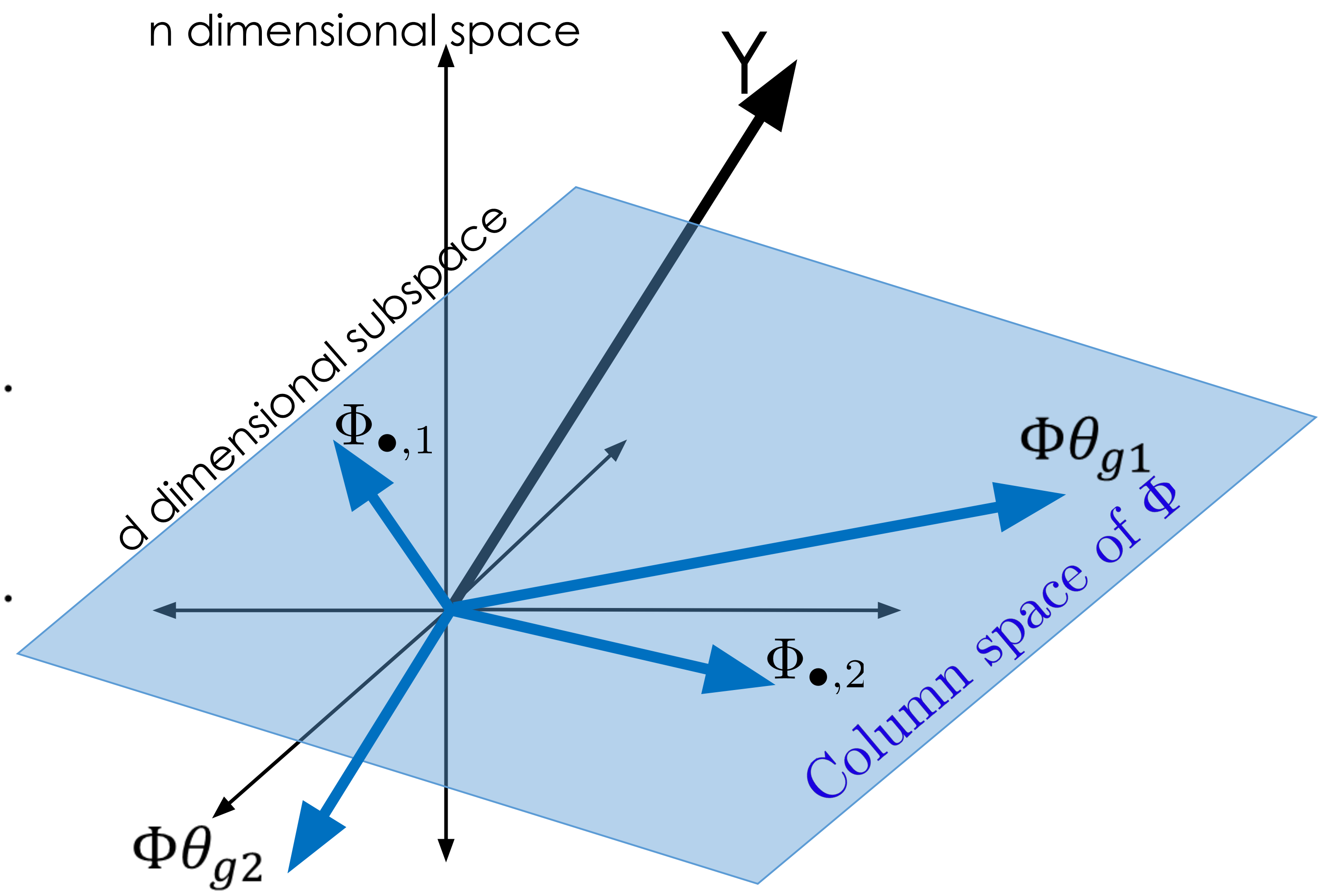
Optimizing the Loss Geometrically

- Our feature matrix Φ has a column space.
  - Can think of this as the set of possible predictions for our N people given the data we have about each.
  - This subspace is d-dimensional.
  - For example, columns could be calorie intake and minutes exercised per day.



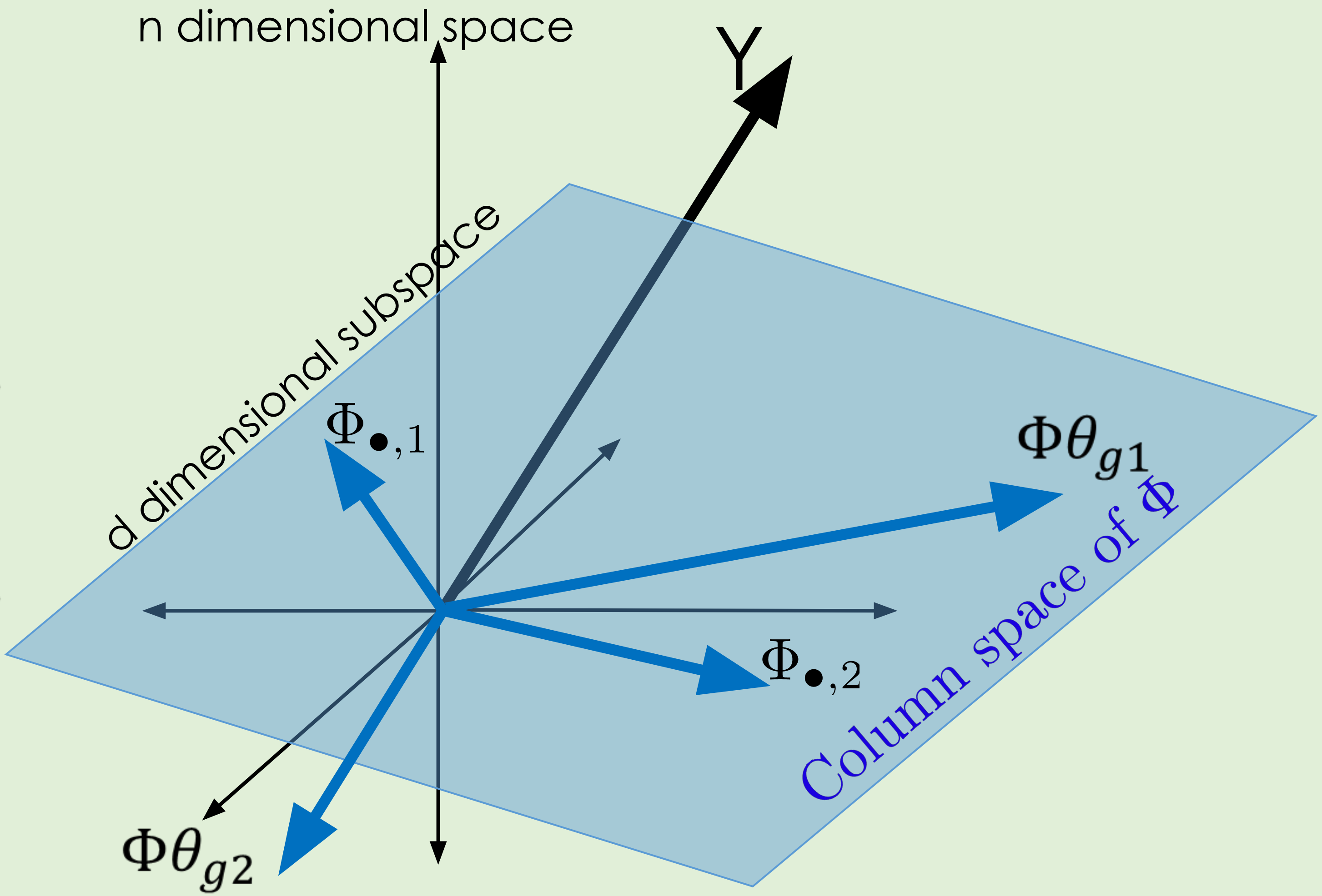
Optimizing the Loss Geometrically

- Picking a parameter vector $\hat{\theta}$ is tantamount to making a prediction for every person.
  - $\Phi\theta_{g1}$ is effectively a prediction for all N people using guess #1.
  - $\Phi\theta_{g2}$ is effectively a prediction for all N people using guess #2.



Optimizing the Loss Geometrically

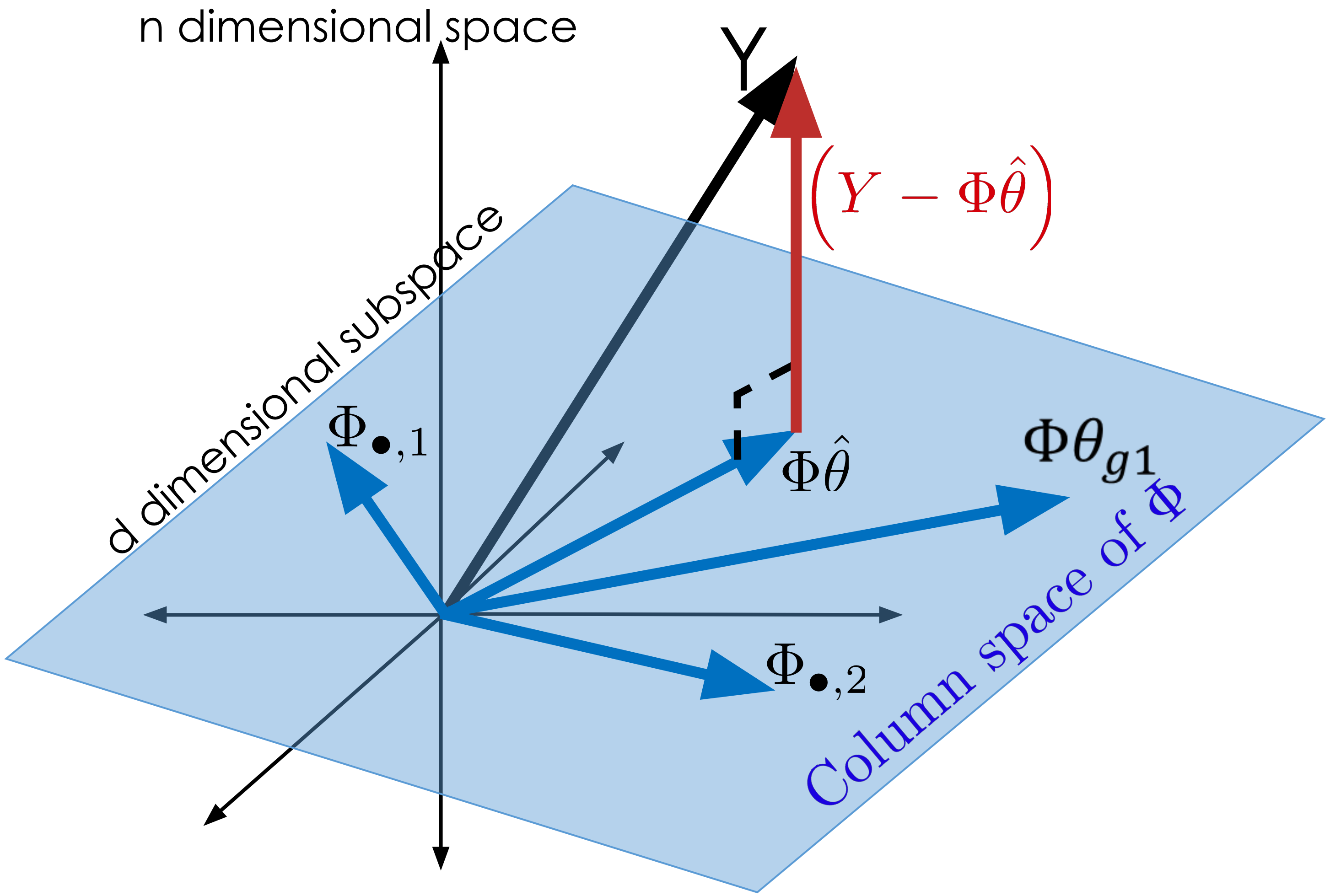
- Picking a parameter vector $\hat{\theta}$ is tantamount to making a prediction for every person.
  - $\Phi\theta_{g1}$ is effectively a prediction for all N people using guess #1.
  - $\Phi\theta_{g2}$ is effectively a prediction for all N people using guess #1.
  - **Which guess is better?**
  - **Where is the optimal solution?**



Optimizing the Loss Geometrically

- The best guess $\hat{\theta}$ minimizes the length of e, where $e = Y - \Phi\hat{\theta}$.
  - e is called the residual.
- This length is minimized if $\Phi\hat{\theta}$ is the projection of Y onto the subspace!
  - In other words, if the residual is orthogonal to the basis vectors of the subspace, then $\hat{\theta}$ is optimal.

More on this in discussion!

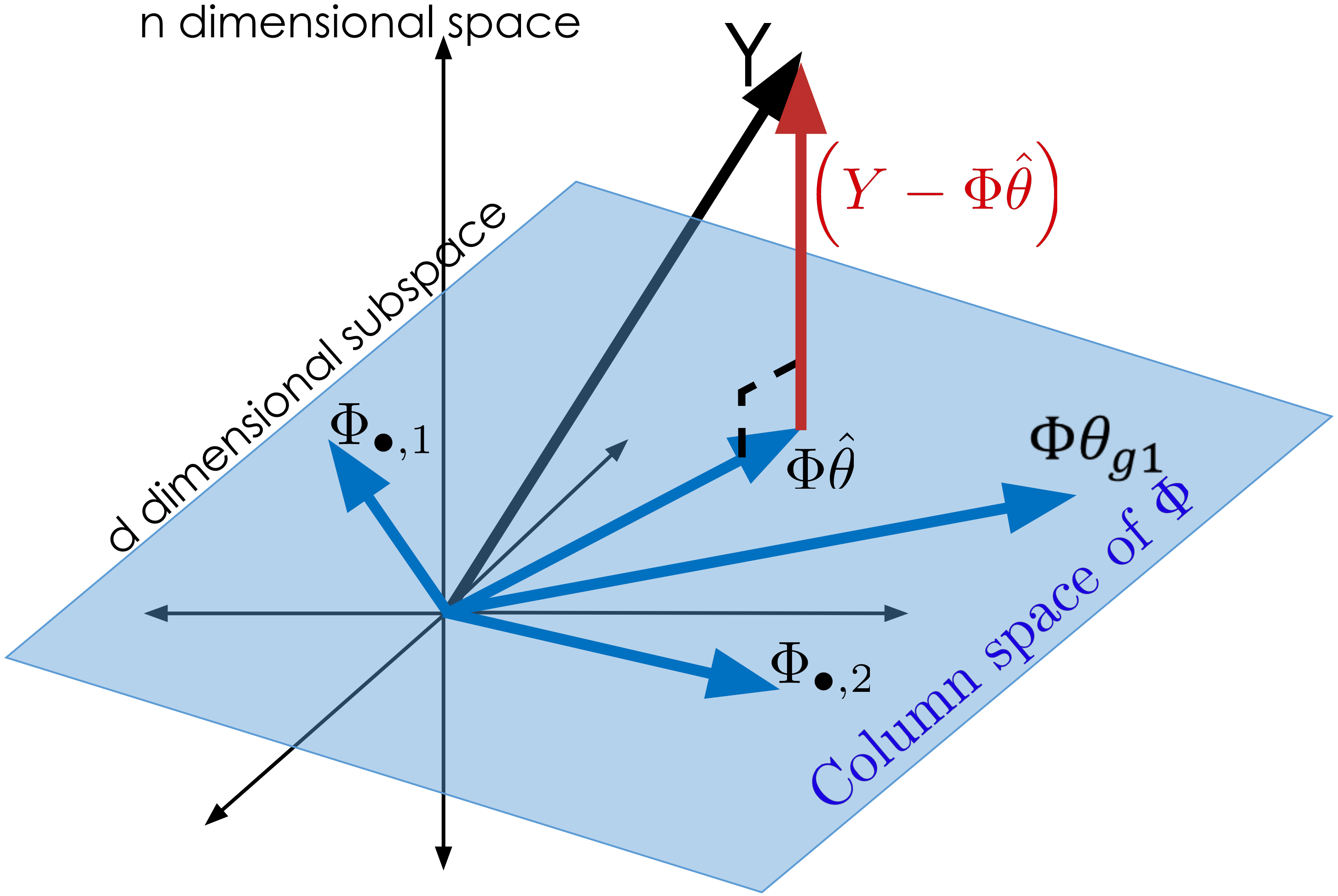


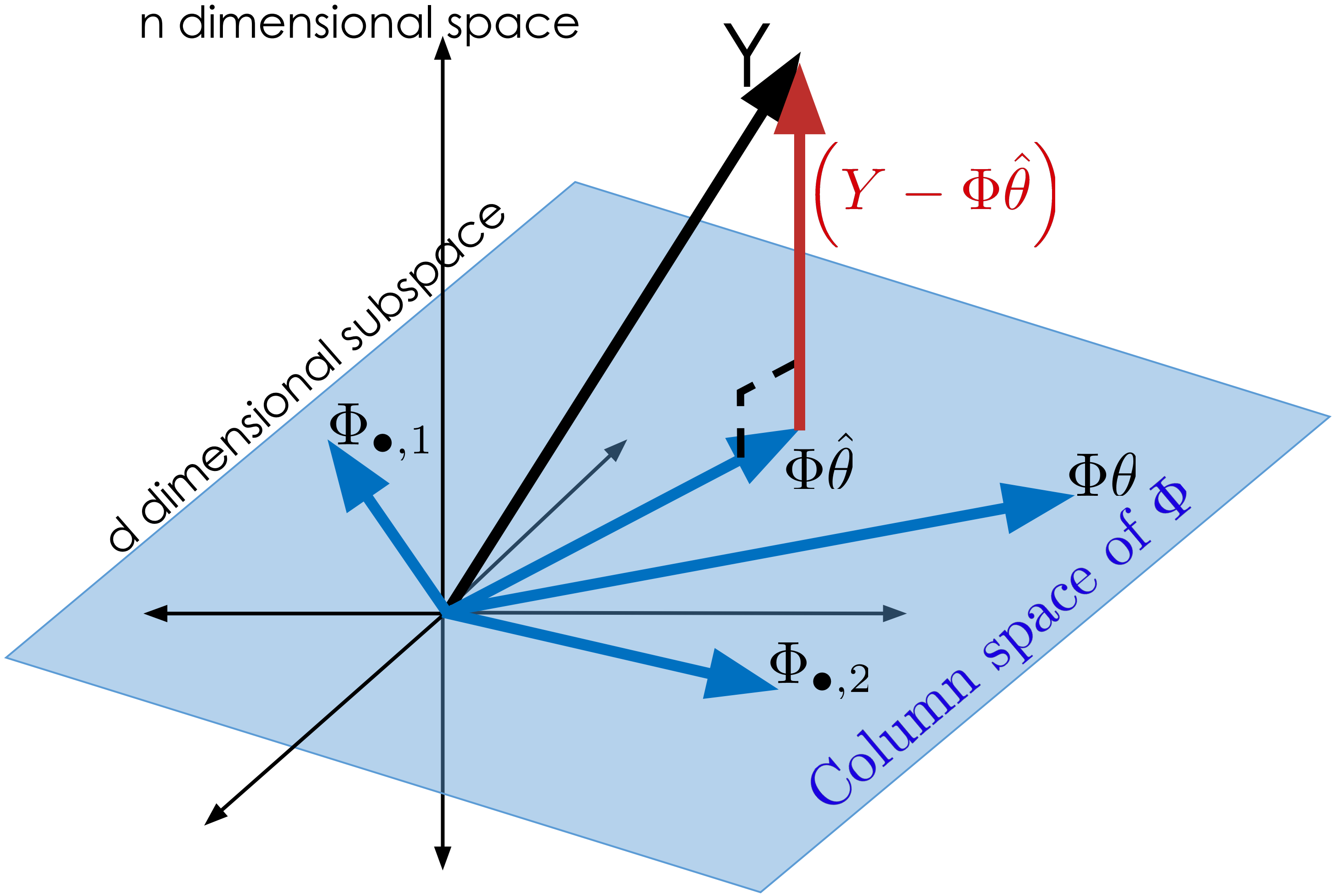
Optimizing the Loss Geometrically

- In other words, if the residual is orthogonal to the basis vectors of the subspace, then $\hat{\theta}$ is optimal. So we need:
  - $0 = \left(Y - \Phi\hat{\theta}\right) \cdot \Phi_{\bullet,1}$
  - $0 = \left(Y - \Phi\hat{\theta}\right) \cdot \Phi_{\bullet,2}$
  - …
  - $0 = \left(Y - \Phi\hat{\theta}\right) \cdot \Phi_{\bullet,d}$

Or more simply:

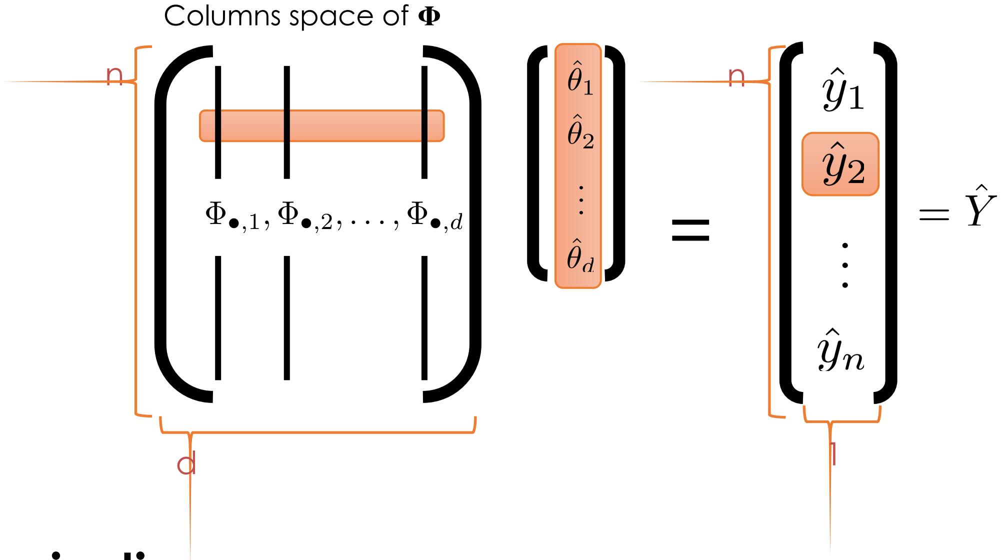
$$0 = \Phi^T\left(Y - \Phi\hat{\theta}\right)$$





Definition of orthogonality

$$0 = \underbrace{\Phi^T}_{\text{d x n}}\underbrace{(Y - \Phi\hat{\theta})}_{\text{n x 1 residual}}$$



Derivation

$$0 = \Phi^T \left(Y - \Phi\hat{\theta}\right)$$

$$0 = \Phi^T Y - \Phi^T \Phi\hat{\theta}$$

$$\Phi^T \Phi\hat{\theta} = \Phi^T Y$$

$$\hat{\theta} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y$$

"Normal Equation"

The Normal Equation $\hat{\theta} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y$

Optimizing Linear models is therefore very easy:

- Given Φ and Y, compute $(\Phi^T \Phi)^{-1} \Phi^{\mathrm{T}} \mathrm{Y}$ and you're done.

Note: For $(\Phi^T \Phi)^{-1}$ to exist Φ needs to be full column rank.

- No collinear columns.
- Why? Prove yourself, or see https://www.youtube.com/watch?v=ESSMQH6Y5OA.
- Don't have full rank? Add regularization (see D100).