ECS 171: Machine Learning

Summer 2023

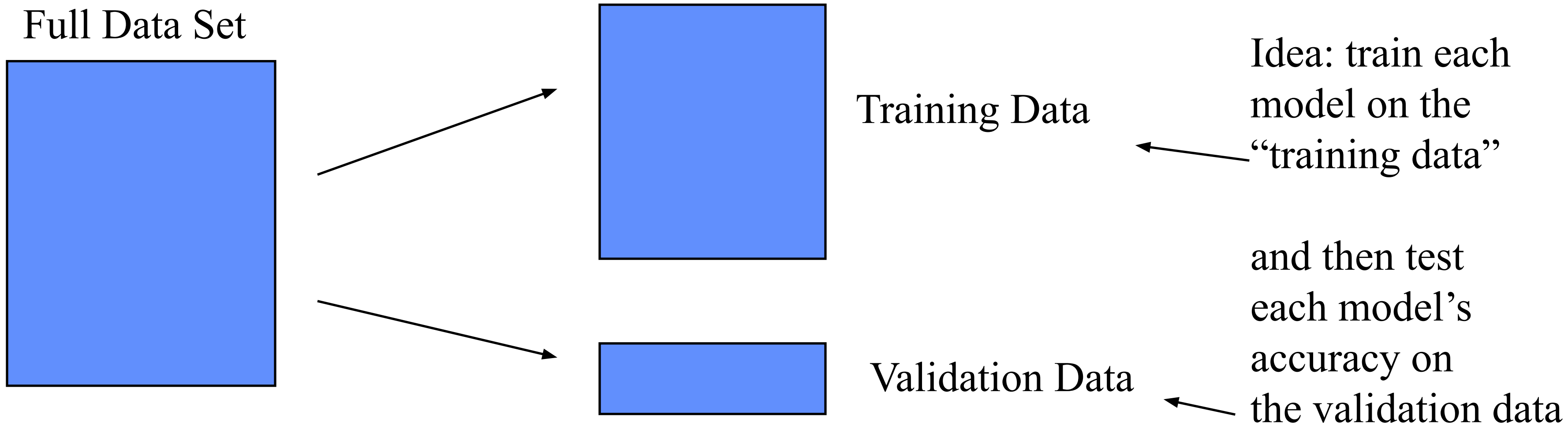
Edwin Solares
easolares@ucdavis.edu

Ordinary Least Squares & Gradient Descent

Remember: Rest & Sleep

Simulated Datasets

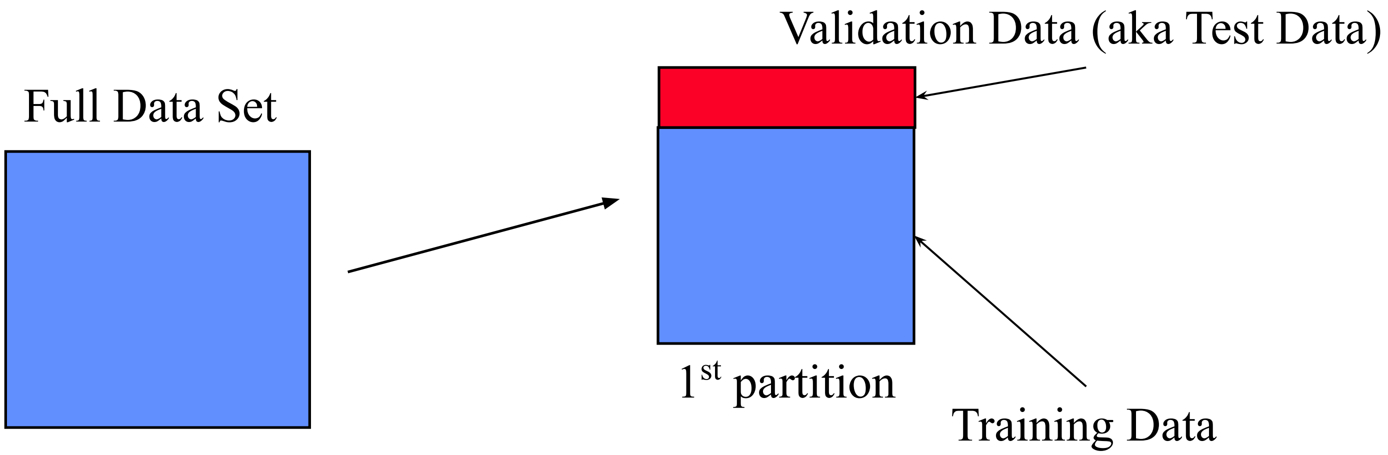
Cross Validation



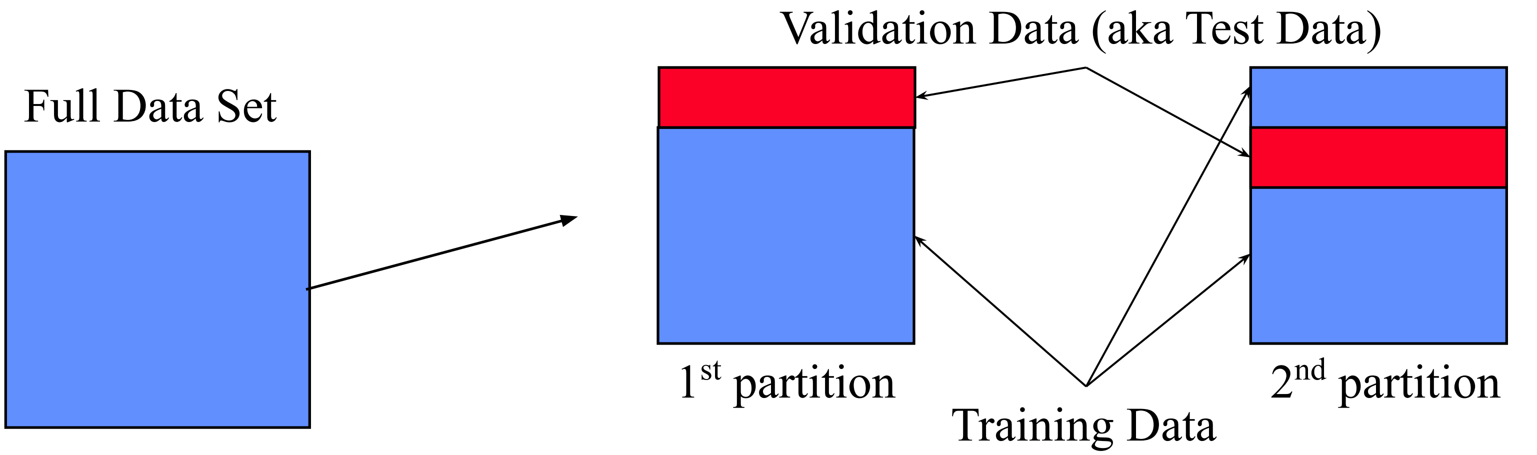
Training data performance is typically optimistic

- e.g., error rate on training data
- build a model on the training data
- assess performance on the test data

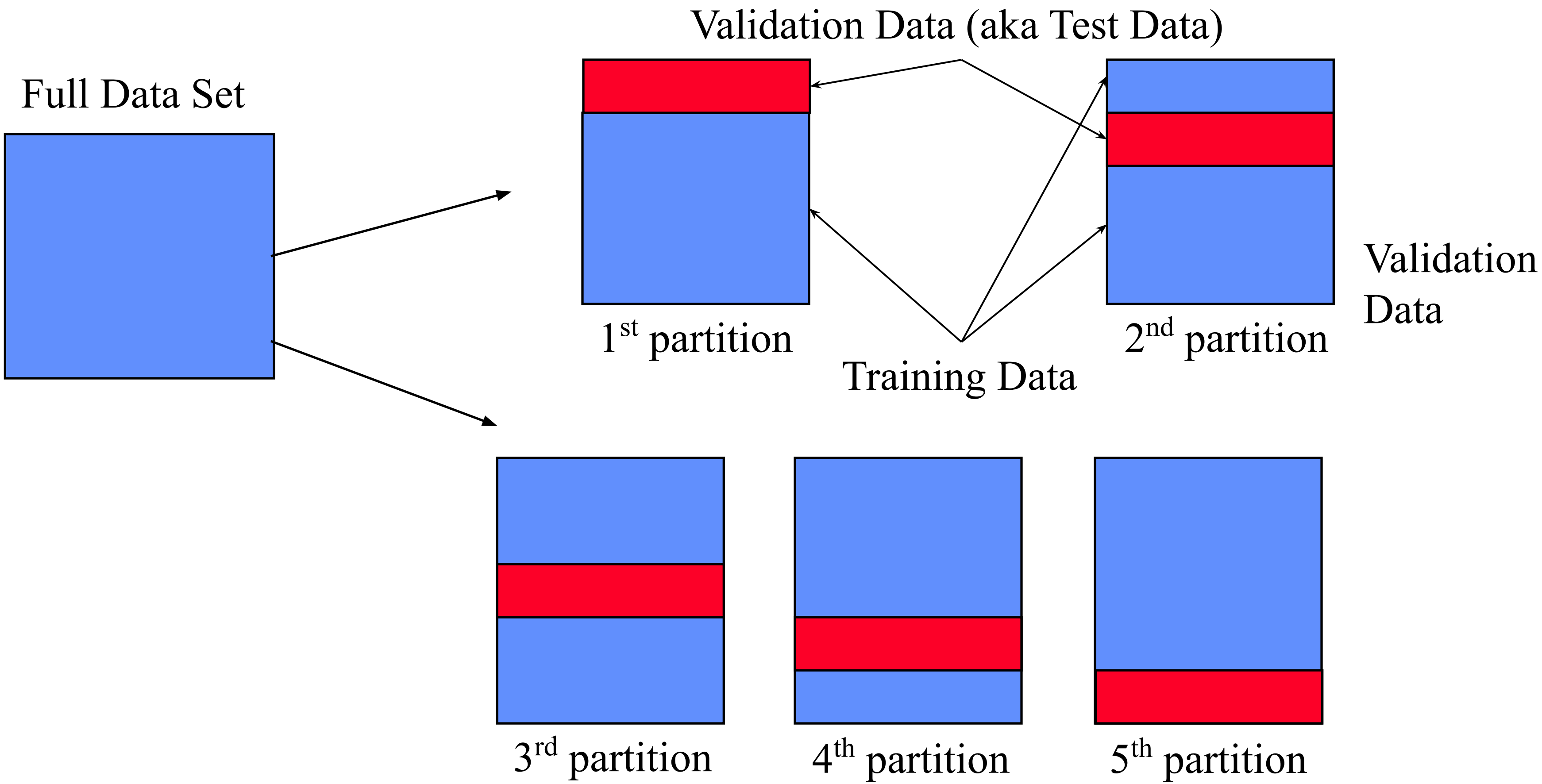
Disjoint Validation Data Sets for k = 5



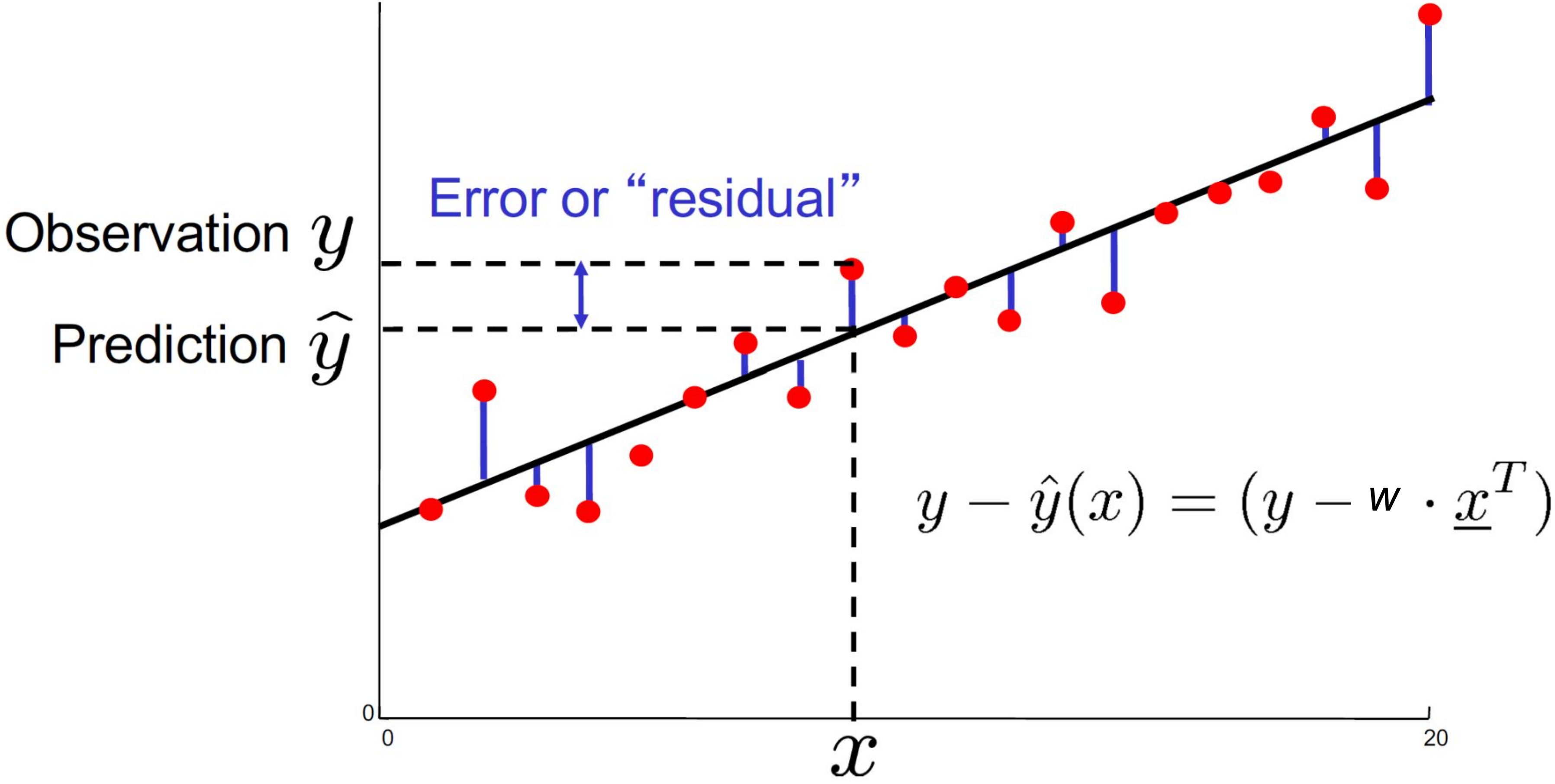
Disjoint Validation Data Sets for k = 5



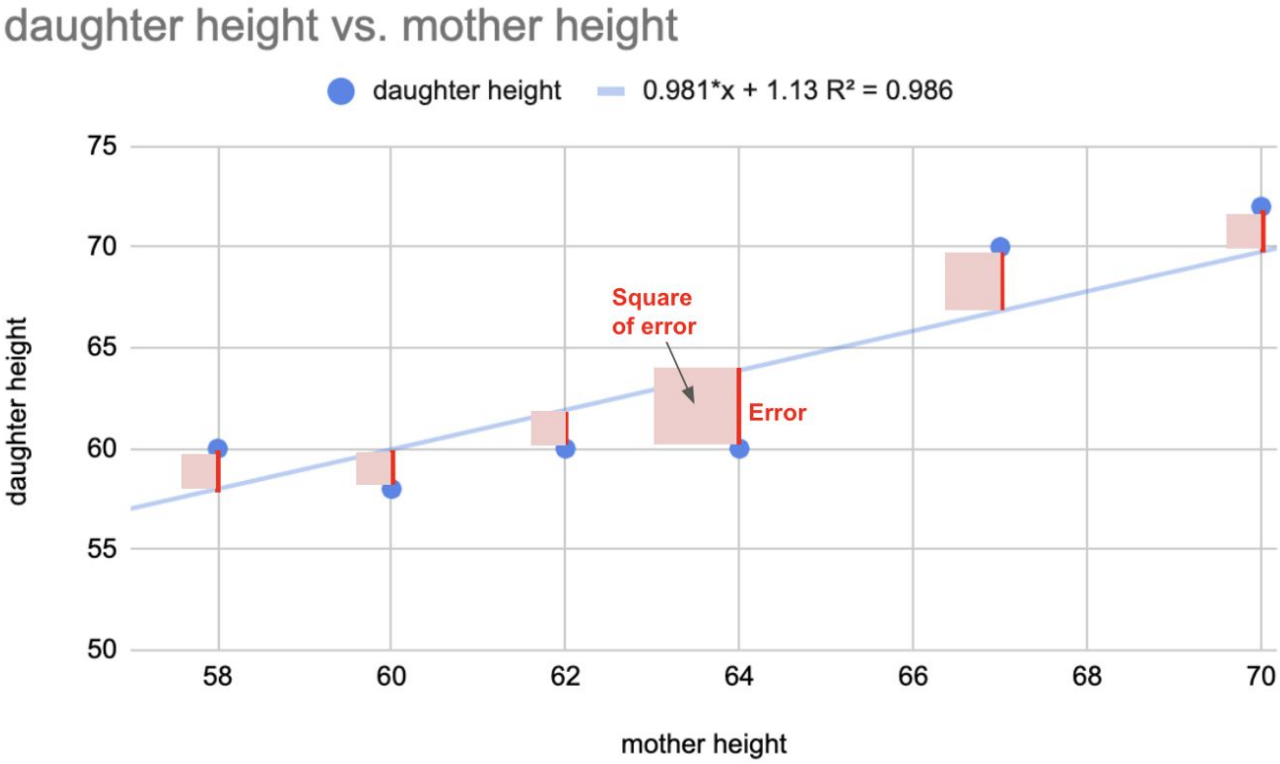
Disjoint Validation Data Sets for k = 5



Remember: Calculating error



Square of the Error Visual



Properties to remember

$$\begin{bmatrix} 1 \\ x_{i,1} \\ x_{i,2} \\ x_{i,3} \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} 1 & x_{i,1} & x_{i,2} & x_{i,3} \end{bmatrix}$$

$$\sum_{j=1}^{m} x_{i,j}^2 = x_i^{\mathsf{T}} \cdot x_i$$

$$\sum_{j=1}^{m} x_{i,j} w_{i,j} = x_i^{\mathsf{T}} \cdot w_i$$

$$x_i^{\mathsf{T}} \cdot w = \hat{y}_i$$

Visualizing the Math

$$x_i \qquad\qquad w^\mathsf{T} \quad = \quad \hat{y}_i$$

$$\begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,m} \\ 1 & x_{2,1} & \dots & x_{2,m} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n,1} & \dots & x_{n,m} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_n \end{bmatrix}$$

$$\begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,3} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = w_0 + w_1 x_{1,1} + w_2 x_{1,2} + w_3 x_{1,3} = \hat{y}_i$$

Recall

$$RSS = \epsilon^T \epsilon = \sum_{i=1}^{m} (\epsilon_i)^2 = \sum_{i=1}^{m} (y_i - w^\intercal x_i)^2$$

Take the derivative

$$\frac{\delta}{\delta w} \sum_{i=1}^{m} (y_i - x_i w^\intercal)^2 = 0$$

$$2 \sum_{i=1}^{m} x_i (y_i - x_i \hat{w}^\intercal) - (y_i - x_i \hat{w}^\intercal) = 0$$

OLS Method

$$\hat{w} = \frac{\sum_{i=1}^{m} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{m} (x_i - \bar{x})^2}$$

$$\sigma_{xy} = \sum_{i=1}^{m} (y_i - \bar{y})(x_i - \bar{x})$$

$$\sigma_x = \sum_{i=1}^{m} (x_i - \bar{x})$$

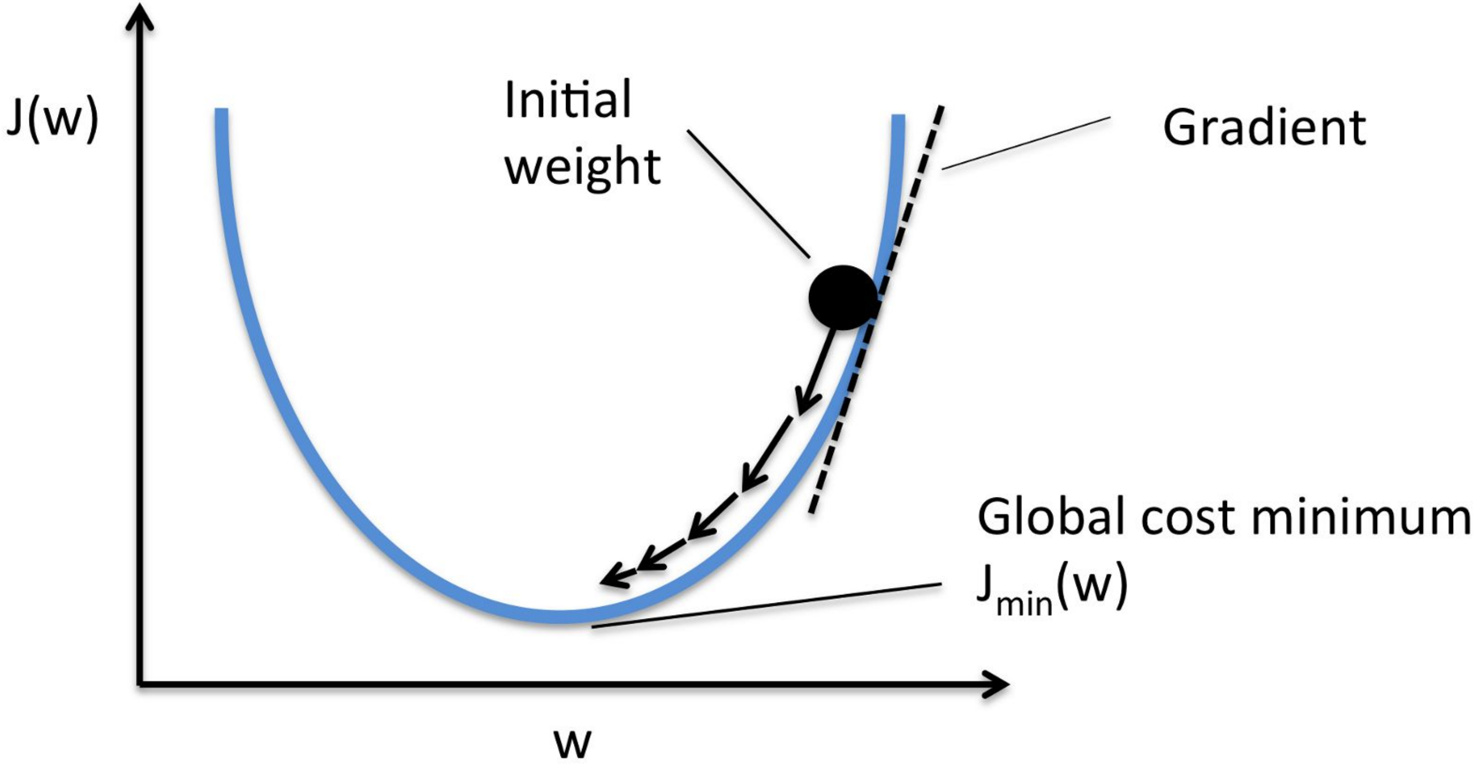
$$\hat{w} = \frac{\sigma_{xy}}{\sigma_x^2}$$

Covariance(x,y)/Variance(x)

approximation

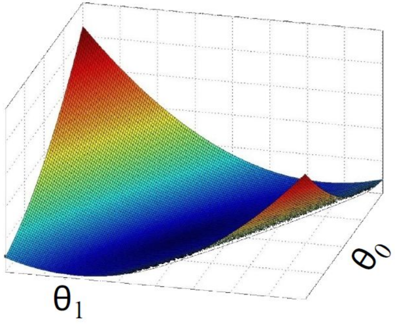
Courtesy of Nvidia Website

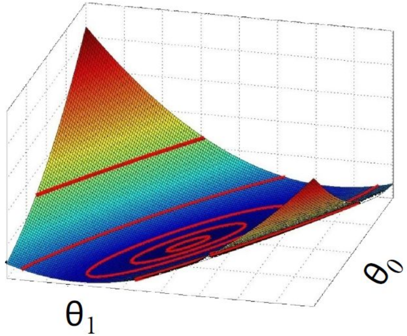
Gradient Descent



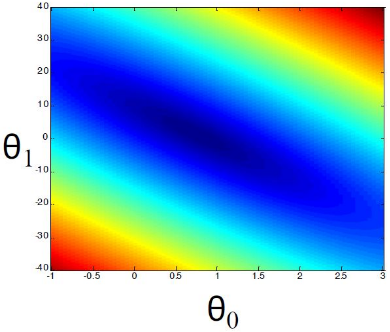
Gradient Descent

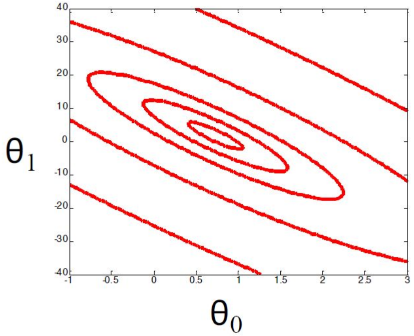
z-axis J(w)





Color for J(w)

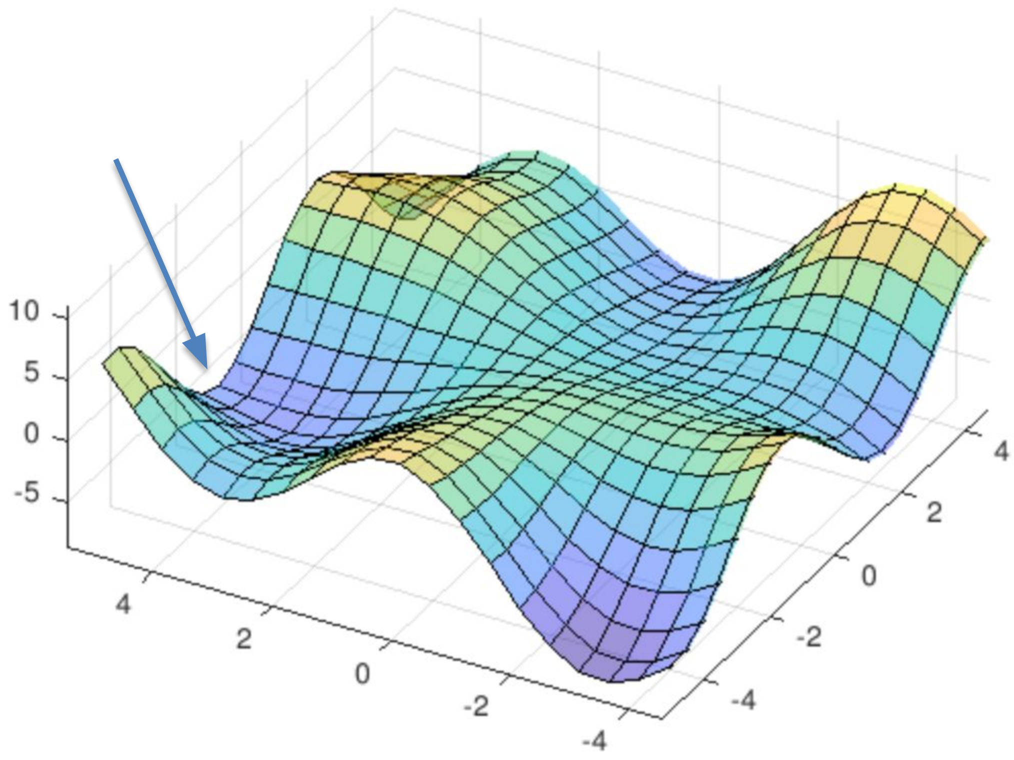




Gradient boundaries for J(w)

Courtesy of Nvidia Website

Gradient Descent



Gradient Descent

$$Loss = \sum_{i=1}^{m} (y_i - x_i w)^2$$

$$\frac{\delta}{\delta w} \sum_{i=1}^{m} (y_i - x_i w)^2 = 0$$

Gradient Descent

$$2\sum_{i=1}^{m} x_i(y_i - x_i\hat{w}) - (y_i - x_i\hat{w}) = 0$$

$$\frac{\delta}{\delta w}\boxed{\frac{1}{2m}}\sum_{i=1}^{n}(y_i - \hat{y})^2 = 0$$

Gradient Descent

$$\sum_{i=1}^{m} x_i(y_i - x_i\hat{w}) - (y_i - x_i\hat{w}) = 0$$

Gradient Descent

$$\frac{1}{m}\sum_{i=1}^{m} x_i(y_i - x_i w) = 0$$

Slope

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - x_i w) = 0$$

y-intercept

Gradient Descent

Define $\alpha =$ Learning rate (step size) for each time point t

We define our GD update function as:

$$J(w_j)_t := J(w_j)_{t-1} - \alpha(\frac{1}{m}\sum_{i=1}^{m}(y_i - x_i w) + \frac{1}{m}\sum_{i=1}^{m} x_i(y_i - x_i w))$$

Gradient Descent

where

$$\alpha(\frac{1}{m}\sum_{i=1}^{m}(y_i - x_i w))$$

defines the y-intercept and

$$\alpha(\frac{1}{m}\sum_{i=1}^{m}x_i(y_i - x_i w))$$

defines the slope

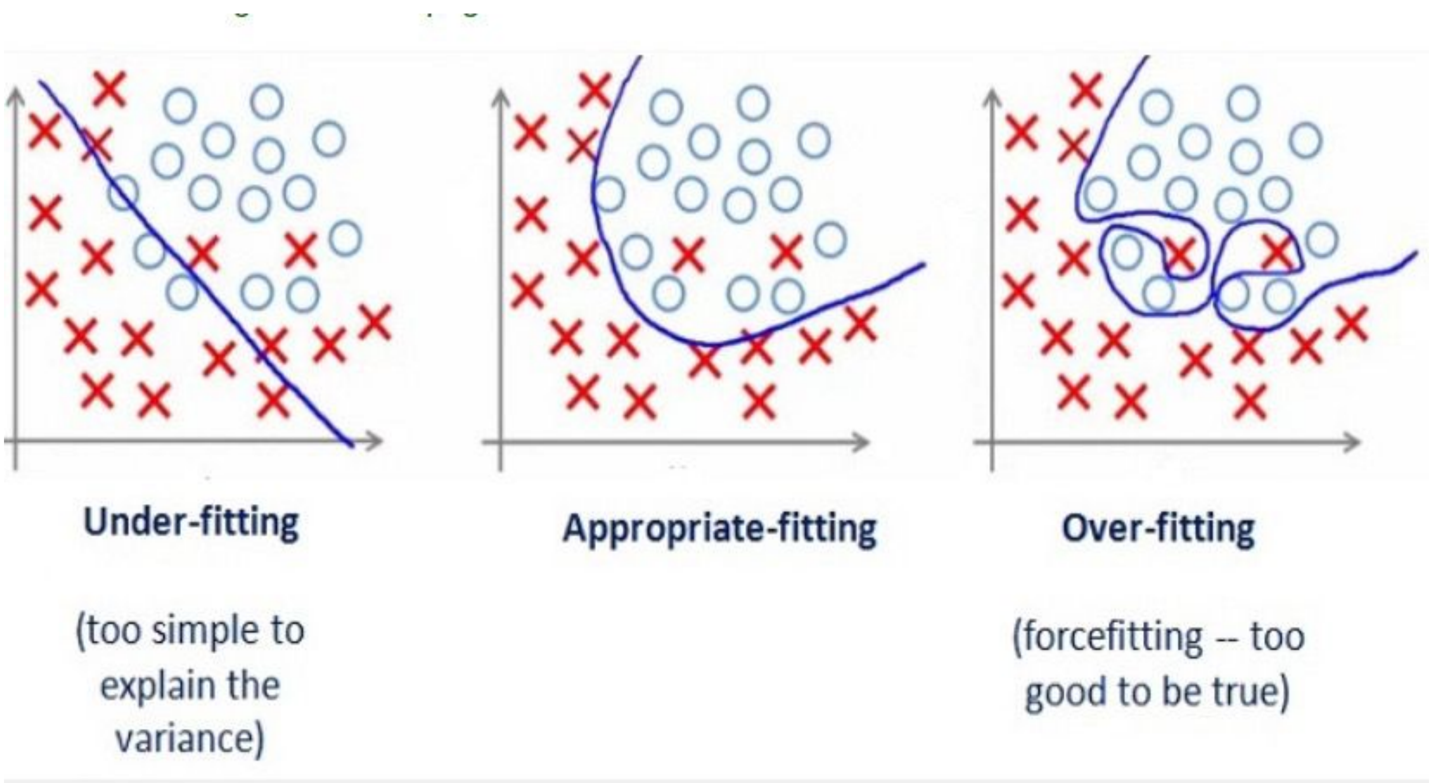
Gradient Descent

https://towardsdatascience.com/step-by-step-tutorial-on-linear-regression-with-stochastic-gradient-descent-1d35b088a843

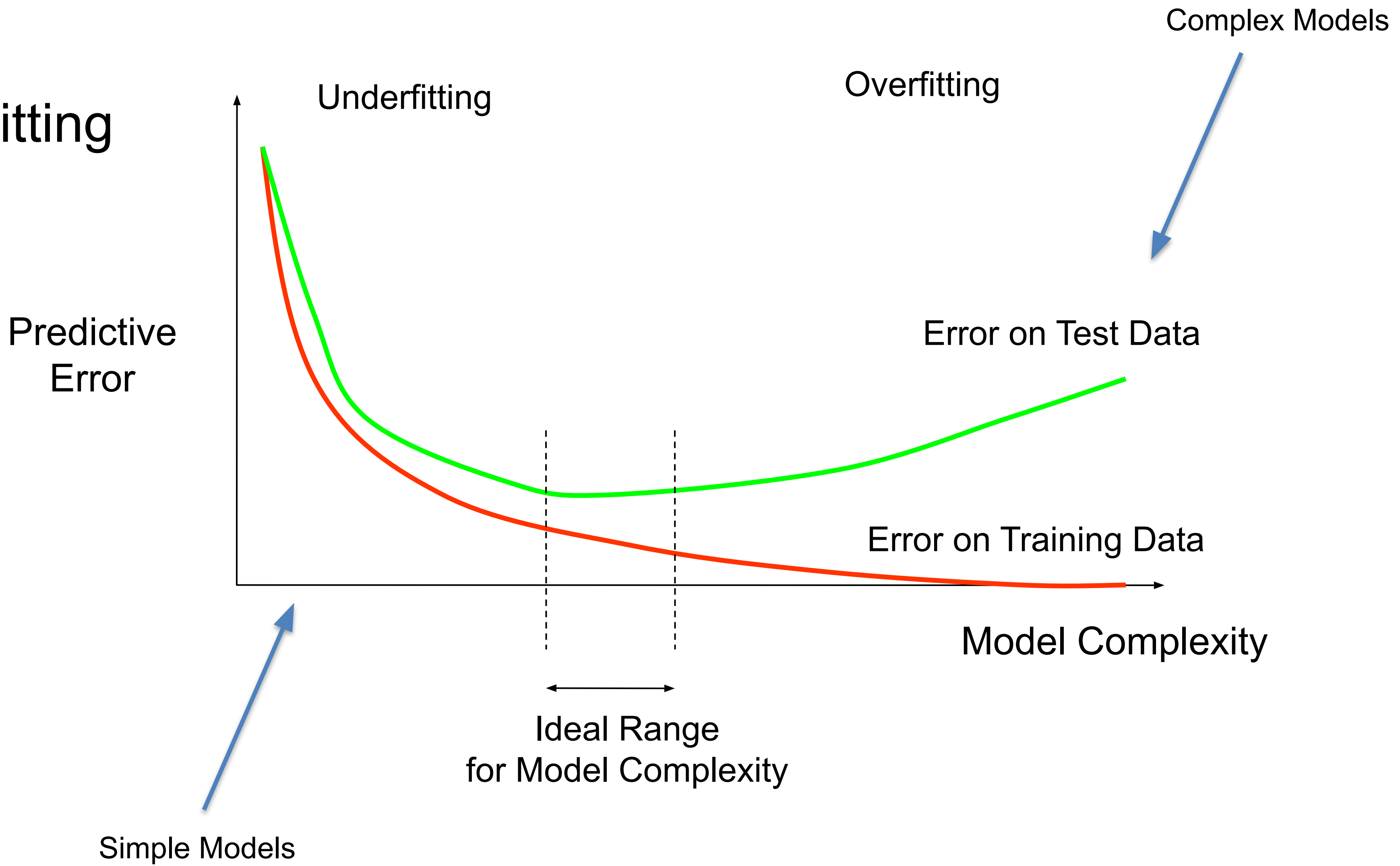
https://towardsdatascience.com/linear-regression-and-gradient-descent-for-absolute-beginners-eef9574eadb0

https://realpython.com/gradient-descent-algorithm-python/

Fitting


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

Fitting



Complex Models
Underfitting
Overfitting
Predictive Error
Error on Test Data
Error on Training Data
Model Complexity
Ideal Range for Model Complexity
Simple Models

Jupyter Notebooks Time!