Big Data Computing

Master's Degree in Computer Science

2022-2023

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

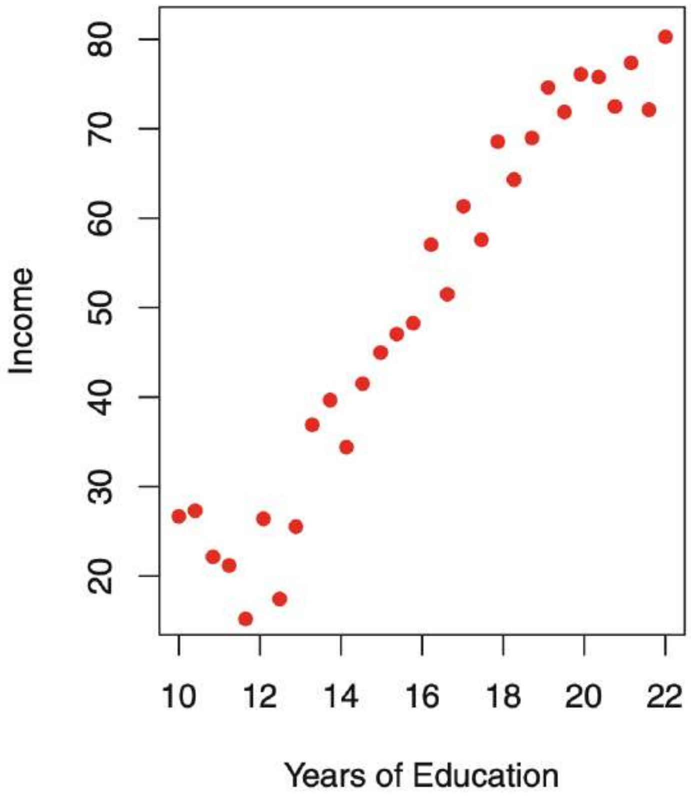
SAPIENZA
UNIVERSITÀ DI ROMA

Recap from Last Lecture

- Supervised Learning as an optimization problem
  - Hypothesis space (assumption)
  - Loss Function (objective)
  - Learning Algorithm (optimizer)
- Regression vs. Classification
- Bias-Variance Tradeoff
- Model selection vs. Model evaluation

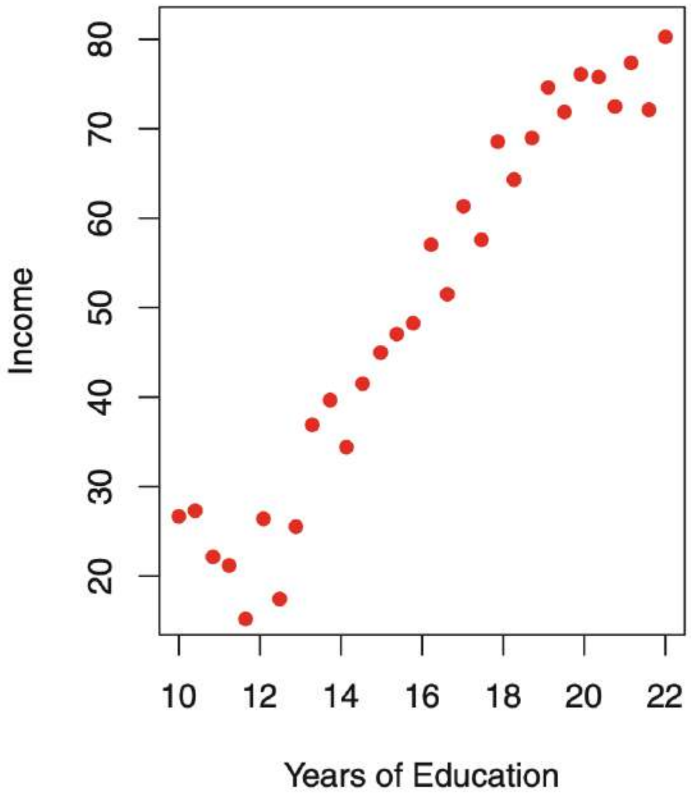
LINEAR REGRESSION

Example: Y=Income vs. X=Education

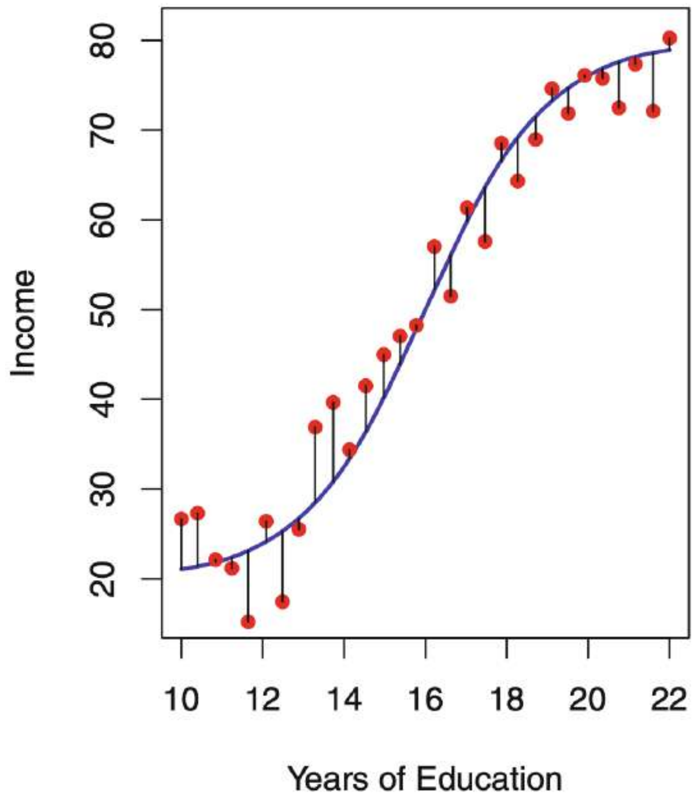


Observations
(simulated)

Example: Y=Income vs. X=Education



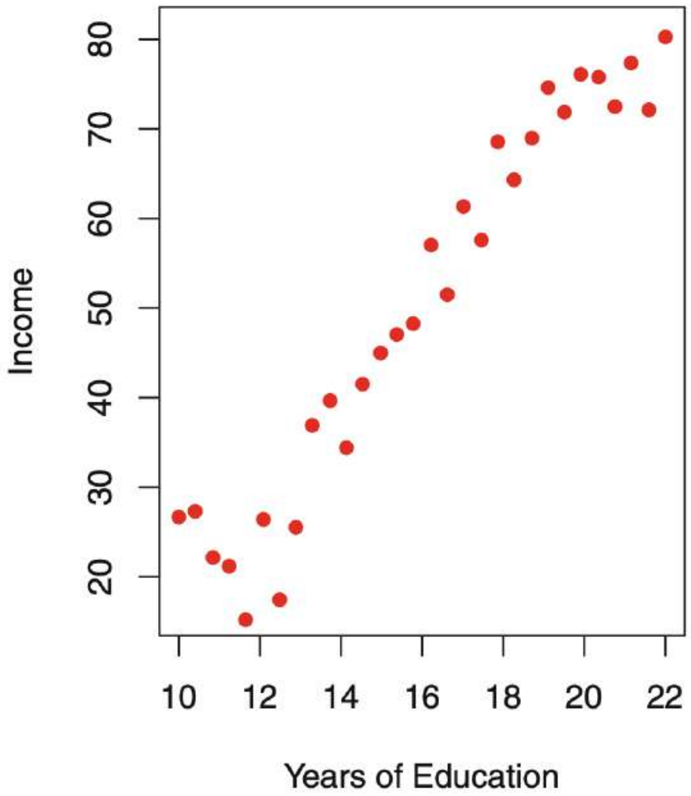
Observations
(simulated)



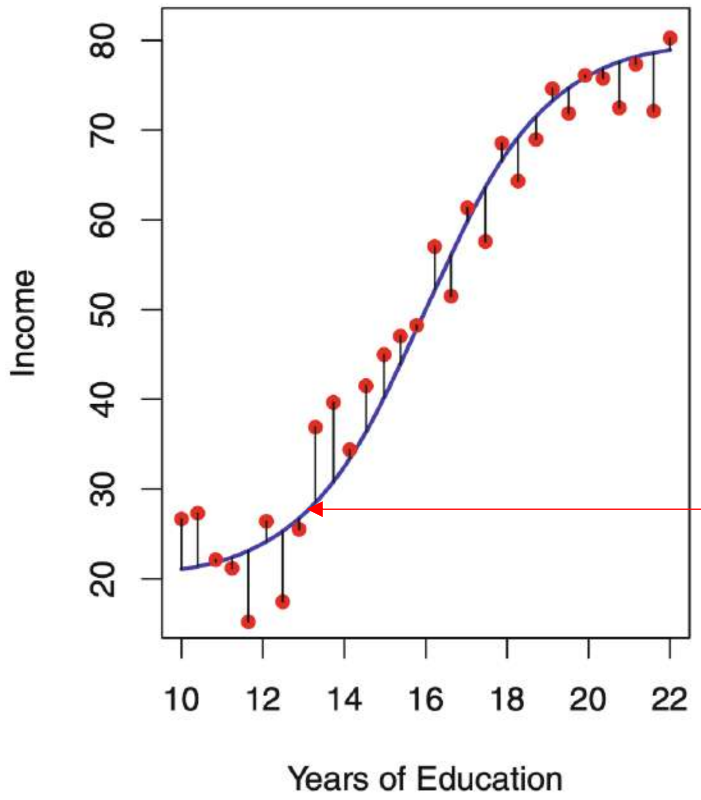
True yet unknown relationship
between X and Y

$$\boxed{Y = f(X) + \varepsilon}$$

Example: Y=Income vs. X=Education



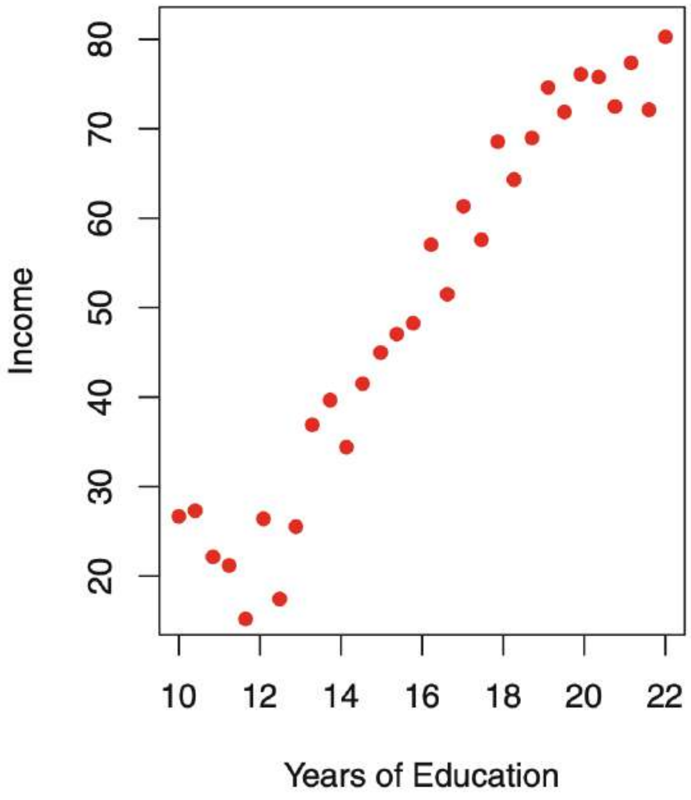
Observations
(simulated)



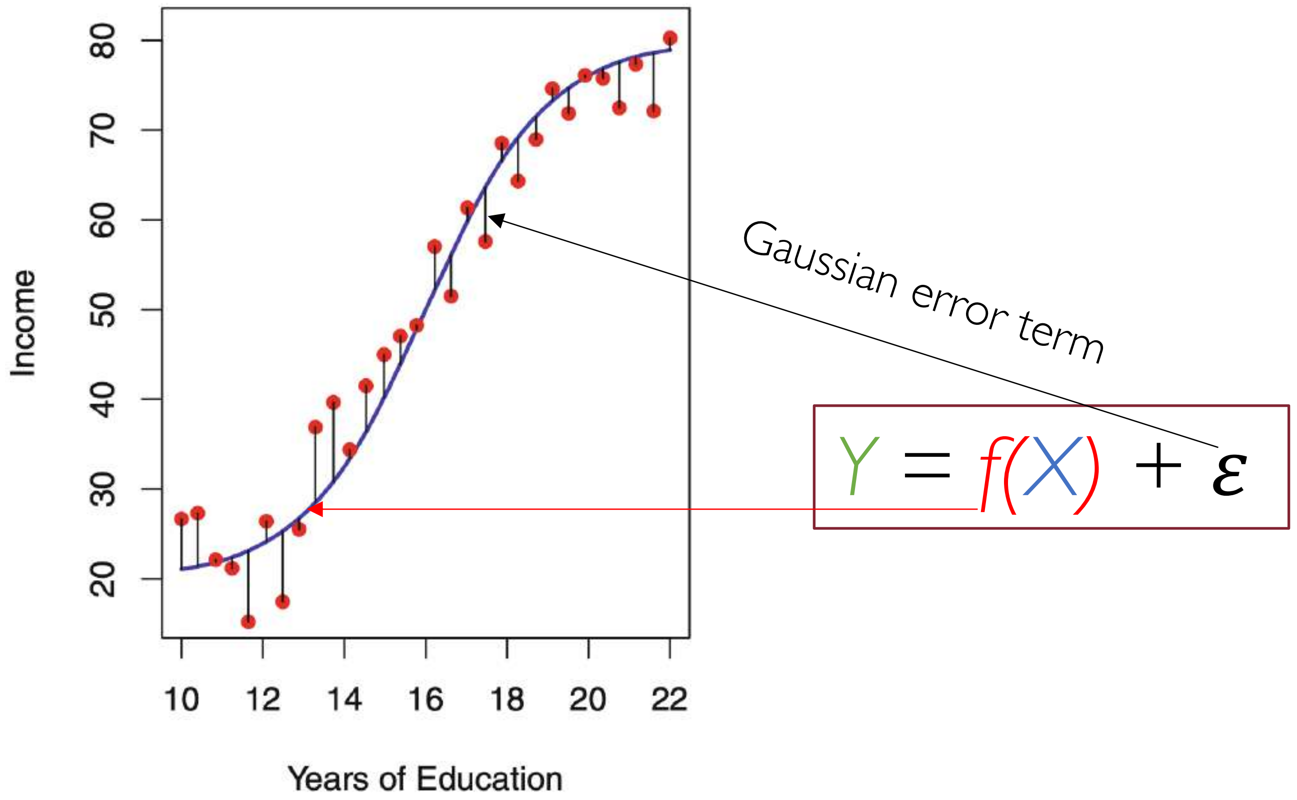
$$Y = f(X) + \varepsilon$$

True yet unknown relationship
between X and Y

Example: Y=Income vs. X=Education



Observations
(simulated)



True yet unknown relationship
between X and Y

Assumptions

- There exists a relationship between X (features) and Y (values)

$$\mathcal{Y} = f(\mathcal{X}) + \epsilon$$

Assumptions

- There exists a relationship between X (features) and Y (values)

$$\mathcal{Y} = f(\mathcal{X}) + \epsilon$$

- f is some fixed but unknown function of X

Assumptions

- There exists a relationship between X (features) and Y (values)

$$\mathcal{Y} = f(\mathcal{X}) + \epsilon$$

- f is some fixed but unknown function of X
- ε is a random error term, which is independent of X and has 0-mean

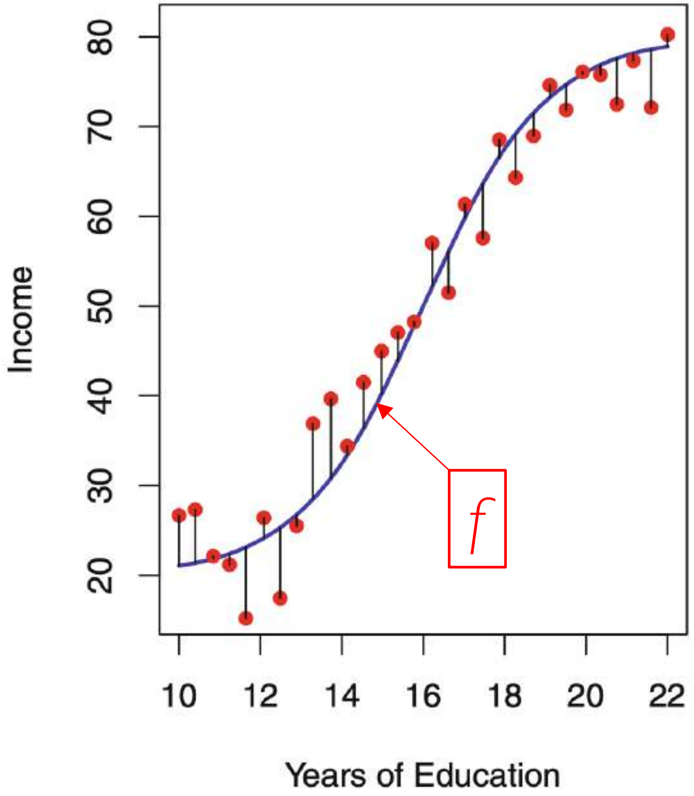
Assumptions

- There exists a relationship between X (features) and Y (values)

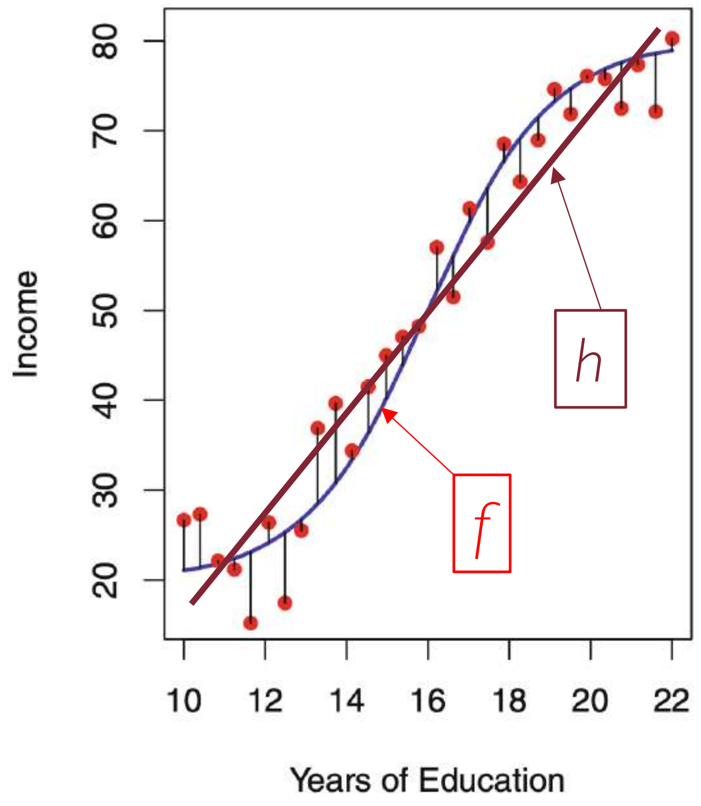
$$\mathcal{Y} = f(\mathcal{X}) + \epsilon$$

- *f* is some fixed but unknown function of X
- ε is a random error term, which is independent of X and has 0-mean
- In this formulation, *f* represents the systematic information that X provides about Y

Goal

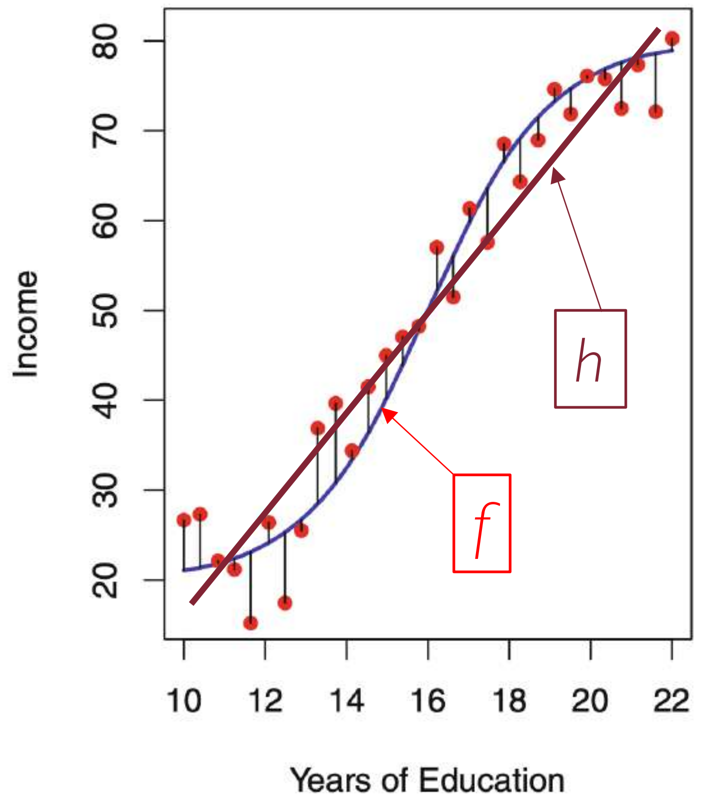

Income
80
70
60
50
40
30
20
10
12
14
16
18
20
22
Years of Education
f

Goal



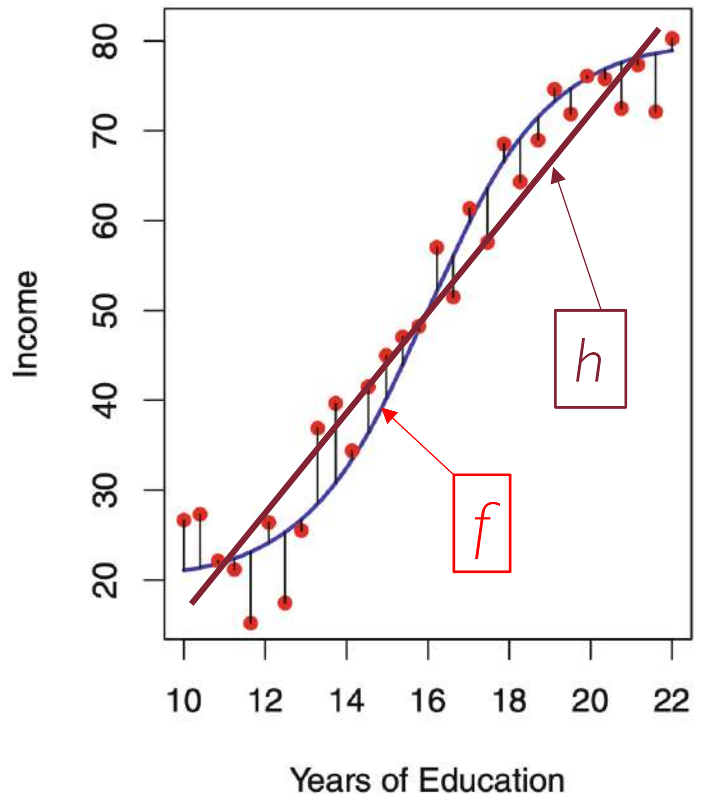
- Find an approximation *h* of the true relationship *f*

Goal



- Find an approximation *h* of the true relationship *f*
- Choose *h* from a specific hypothesis space *H* (i.e., linear functions)

Goal



- Find an approximation h of the true relationship f
- Choose h from a specific hypothesis space H (i.e., linear functions)
- Use a dataset D of observations to learn h

$$\boxed{h(X) \sim f(X)}$$

Recap of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$	input feature space
$\mathcal{Y}$	output space
$\mathcal{Y} \subseteq \mathbb{R}$	real-value label(regression)
$(\mathbf{x}_i, y_i)$	*i*-th labeled instance
$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n}) \in \mathcal{X}$	*n*-dimensional feature vector of the *i*-th instance
$y_i \in \mathcal{Y}$	label of the *i*-th instance
$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$	dataset of *m* i.i.d. labeled instances

The Hypothesis Space *H*

The hypothesis space is defined as follows:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{Y} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n\}$$

The Hypothesis Space *H*

The hypothesis space is defined as follows:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{Y} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n\}$$

$\boldsymbol{\theta}$ $n+1$-dimensional vector of model parameters

The Hypothesis Space *H*

The hypothesis space is defined as follows:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{Y} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n\}$$

$\boldsymbol{\theta}$ $\quad$ $n+1$-dimensional vector of model parameters

$x_0 = 1$ by convention

The Hypothesis Space H

The hypothesis space is defined as follows:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{Y} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n\}$$

$\boldsymbol{\theta}$ $n+1$-dimensional vector of model parameters

$x_0 = 1$ by convention

Among all the possible instantiations of $\boldsymbol{\theta}$ the learning algorithm selects $\boldsymbol{\theta}^*$ as the one which minimizes a loss function measured on D

Residual Sum of Squares (RSS)

$y_i = f(\mathbf{x}_i) + \epsilon_i$ *i*-th observation

Residual Sum of Squares (RSS)

$y_i = f(\mathbf{x}_i) + \epsilon_i$ *i*-th observation

$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ *i*-th prediction

Residual Sum of Squares (RSS)

$y_i = f(\mathbf{x}_i) + \epsilon_i$ *i*-th observation $\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ *i*-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}$$

Residual Sum of Squares (RSS)

$y_i = f(\mathbf{x}_i) + \epsilon_i$ *i*-th observation

$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ *i*-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}$$

$$e_i = \hat{y}_i - y_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \underbrace{y_i}_{f(\mathbf{x}_i)+\epsilon_i}$$ *i*-th residual

Residual Sum of Squares (RSS)

$y_i = f(\mathbf{x}_i) + \epsilon_i$ *i*-th observation

$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ *i*-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}$$

$$e_i = \hat{y}_i - y_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \underbrace{y_i}_{f(\mathbf{x}_i)+\epsilon_i}$$ *i*-th residual

$$\boxed{\mathrm{RSS}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2}$$

Ordinary Least Squares (OLS)

- Remember that the supervised learning problem can be generally defined as the following optimization problem

$$h^* = \text{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})$$

Ordinary Least Squares (OLS)

- Remember that the supervised learning problem can be generally defined as the following optimization problem

$$h^* = \text{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})$$

- OLS is the usual approach to fit (i.e., find the optimal set of parameters of) linear regression models

Ordinary Least Squares (OLS)

- Remember that the supervised learning problem can be generally defined as the following optimization problem

$$h^* = \text{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})$$

- OLS is the usual approach to fit (i.e., find the optimal set of parameters of) linear regression models

$$\boxed{h^* = h_{\boldsymbol{\theta}^*} = \text{argmin}_{\boldsymbol{\theta}} L(h_{\boldsymbol{\theta}}, \mathcal{D})}$$

The Loss Function L: Mean Squared Error

- OLS uses Mean Squared Error as the loss function to minimize

The Loss Function L: Mean Squared Error

- OLS uses Mean Squared Error as the loss function to minimize
- MSE measures the average error when the true f is substituted with a hypothesis $h_{\boldsymbol{\theta}}$ in H (in-sample error)

The Loss Function *L*: Mean Squared Error

- OLS uses Mean Squared Error as the loss function to minimize
- MSE measures the average error when the true f is substituted with a hypothesis $h_{\boldsymbol{\theta}}$ in H (in-sample error)

$$\mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{1}{m}\mathrm{RSS}(h_{\boldsymbol{\theta}}, \mathcal{D}) =$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2$$

The OLS Learning Algorithm

OLS aims at solving the following optimization problem:

$$h^* = h_{\boldsymbol{\theta}^*} = \text{argmin}_{\boldsymbol{\theta}} \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) =$$

$$= \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

The OLS Learning Algorithm

OLS aims at solving the following optimization problem:

$$h^* = h_{\boldsymbol{\theta}^*} = \text{argmin}_{\boldsymbol{\theta}} \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) =$$

$$= \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

How do we solve that?

The OLS Learning Algorithm

OLS aims at solving the following optimization problem:

$$h^* = h_{\boldsymbol{\theta}^*} = \text{argmin}_{\boldsymbol{\theta}} \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) =$$

$$= \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

NOTE:

The function to minimize can be proven convex

Min/Max of a Convex/Concave Function

- Any local minimum (maximum) of a convex (concave) function is also a global minimum (maximum)

Min/Max of a Convex/Concave Function

- Any local minimum (maximum) of a convex (concave) function is also a global minimum (maximum)
- If the function is convex (concave) finding the global minimum (maximum) can be done just by computing the first derivative and set it to 0

Min/Max of a Convex/Concave Function

- Any local minimum (maximum) of a convex (concave) function is also a global minimum (maximum)
- If the function is convex (concave) finding the global minimum (maximum) can be done just by computing the first derivative and set it to 0
- In the case of a multivariate function, this generalizes to compute the gradient (∇) of the function and set it to 0

The Gradient ∇

The gradient of an *n*-variable function is the *n*-dimensional vector of the partial derivatives of the function w.r.t. each of its variable

The Gradient ∇

The gradient of an *n*-variable function is the *n*-dimensional vector of the partial derivatives of the function w.r.t. each of its variable

$$f : \mathbb{R}^n \mapsto \mathbb{R} \qquad \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

The Gradient ∇

The gradient of an *n*-variable function is the *n*-dimensional vector of the partial derivatives of the function w.r.t. each of its variable

$$f : \mathbb{R}^n \mapsto \mathbb{R} \qquad \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

Solving $\nabla f = 0$ means finding the *n*-dimensional vector **x** such that:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right) = \underbrace{(0, 0, \ldots, 0)}_{n} = \mathbf{0}$$

Solving the Optimization Problem

$$\text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Solving the Optimization Problem

$$\text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Observations y_i and features x_i can be thought of as fixed constants

Solving the Optimization Problem

$$\text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Observations y_i and features x_i can be thought of as fixed constants

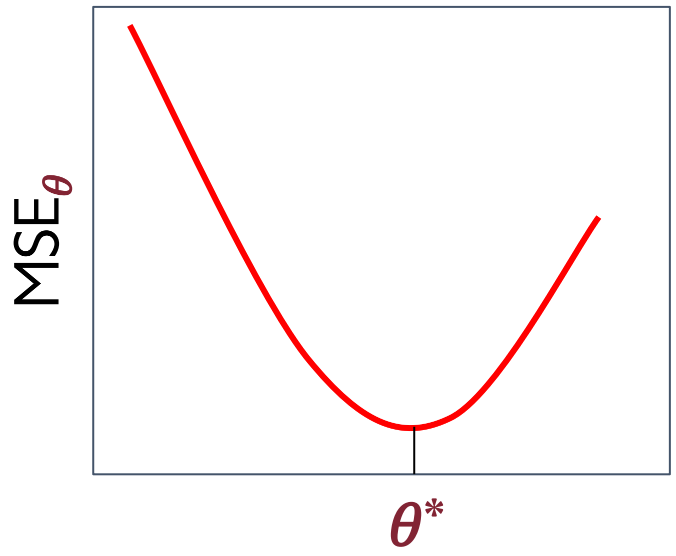
Each term of the summation is a multivariate linear function of the model parameters θ

Solving the Optimization Problem

$$\operatorname{argmin}_{\boldsymbol{\theta}}\left[\frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2\right]$$

Observations y_i and features x_i can be thought of as fixed constants

Each term of the summation is a multivariate linear function of the model parameters θ



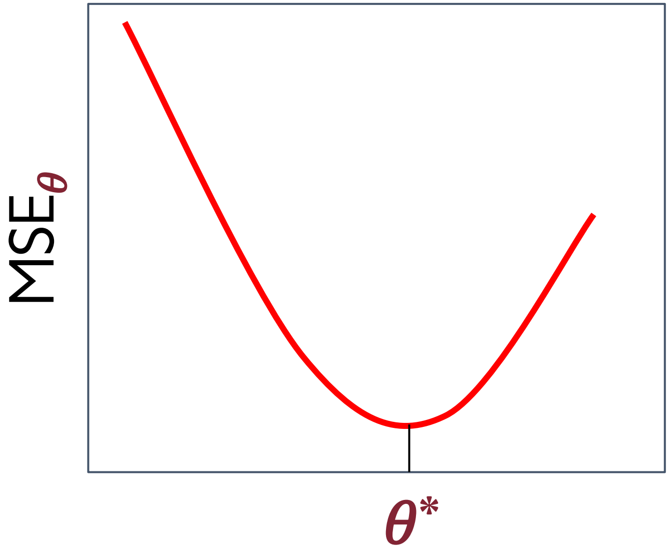
Linear functions are convex and so is any sum of those

Solving the Optimization Problem

$$\text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Observations y_i and features x_i can be thought of as fixed constants

Each term of the summation is a multivariate linear function of the model parameters θ



Linear functions are convex and so is any sum of those

Convex functions have a unique local minimum, which therefore happens to be the global minimum

Computing the Gradient of MSE

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Computing the Gradient of MSE

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

$$\boxed{\frac{\partial f}{\partial t}(\alpha t) = \alpha \frac{\partial f}{\partial t}(t) \;\; \alpha \in \mathbb{R}, \text{constant}}$$

scalar multiple rule

Computing the Gradient of MSE

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

$$\boxed{\frac{\partial f}{\partial t}(\alpha t) = \alpha \frac{\partial f}{\partial t}(t) \;\; \alpha \in \mathbb{R}, \text{constant}}$$

scalar multiple rule

$$\boxed{\frac{\partial f}{\partial t}\left(\sum t\right) = \left(\sum \frac{\partial f}{\partial t}(t)\right)}$$

sum rule

Computing the Gradient of MSE

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla \left[\frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

$$\boxed{\frac{\partial}{\partial t} f(\alpha t) = \alpha \frac{\partial}{\partial t} f(t), \alpha \in \mathbb{R} \text{ (constant)}}$$

scalar multiple rule

$$\boxed{\frac{\partial}{\partial t} f\left(\sum t\right) = \sum \left(\frac{\partial}{\partial t} f(t)\right)}$$

sum rule

$$\Downarrow$$

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{1}{m} \left[\sum_{i=1}^{m} \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2 \right]$$

Computing the Gradient of MSE (1 instance)

To make things easier, let's assume the dataset D contains a single instance $(\mathbf{x}, y)$

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)^2$$

Computing the Gradient of MSE (1 instance)

To make things easier, let's assume the dataset *D* contains a single instance (x, *y*)

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)^2$$

$$\boxed{\frac{\partial}{\partial t} t^n = n t^{n-1}, \; n \in \mathbb{N}}$$

power rule

Computing the Gradient of MSE (1 instance)

To make things easier, let's assume the dataset D contains a single instance ($\mathbf{x}$, y)

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)^2$$

$$\boxed{\frac{\partial}{\partial t} t^n = n t^{n-1}, \;\; n \in \mathbb{N}}$$

power rule

$$\boxed{\frac{\partial}{\partial t} f(g(t)) = \frac{\partial}{\partial g(t)} f(g(t)) * \frac{\partial}{\partial t} g(t)}$$

chain rule

Computing the Gradient of MSE (1 instance)

To make things easier, let's assume the dataset D contains a single instance ($\mathbf{x}$, y)

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)^2$$

$$\boxed{\frac{\partial}{\partial t} t^n = n t^{n-1}, \;\; n \in \mathbb{N}}$$

power rule

$$\boxed{\frac{\partial}{\partial t} f(g(t)) = \frac{\partial}{\partial g(t)} f(g(t)) * \frac{\partial}{\partial t} g(t)}$$

chain rule

$$\Downarrow$$

$$2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y) \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)$$

Computing the Gradient of MSE (1 instance)

To make things easier, let's assume the dataset D contains a single instance ($\mathbf{x}$, y)

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)^2$$

$$\boxed{\frac{\partial}{\partial t} t^n = n t^{n-1}, \;\; n \in \mathbb{N}}$$

power rule

$$\boxed{\frac{\partial}{\partial t} f(g(t)) = \frac{\partial}{\partial g(t)} f(g(t)) * \frac{\partial}{\partial t} g(t)}$$

chain rule

$$2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y)\boxed{\nabla (h_{\boldsymbol{\theta}}(\mathbf{x}) - y)}$$

Computing the Gradient of MSE (1 instance)

$$\nabla(h_{\boldsymbol{\theta}}(\mathbf{x}) - y) = \nabla(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y) =$$

Computing the Gradient of MSE (1 instance)

$$\nabla(h_{\boldsymbol{\theta}}(\mathbf{x}) - y) = \nabla(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y) =$$

$$= \Bigg(\underbrace{\frac{\partial(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y)}{\partial \theta_0}, \ldots, \frac{\partial(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y)}{\partial \theta_n}}_{n+1}\Bigg) = (x_0, x_1, \ldots, x_n) = \mathbf{x}$$

Computing the Gradient of MSE (1 instance)

$$\nabla(h_{\boldsymbol{\theta}}(\mathbf{x}) - y) = \nabla(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y) =$$

$$= \Bigg(\underbrace{\frac{\partial(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y)}{\partial\theta_0}, \ldots, \frac{\partial(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y)}{\partial\theta_n}}_{n+1}\Bigg) = (x_0, x_1, \ldots, x_n) = \mathbf{x}$$

$$\Downarrow$$

$$\nabla\mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \underbrace{2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y)}_{\text{scalar}} \cdot \underbrace{\mathbf{x}}_{(n+1)\text{-dimensional vector}}$$

Vectorized Notation

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \underbrace{2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y)}_{\text{scalar}} \cdot \underbrace{\mathbf{x}}_{(n+1)\text{-dimensional vector}}$$

Vectorized Notation

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \underbrace{2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y)}_{\text{scalar}} \cdot \underbrace{\mathbf{x}}_{(n+1)\text{-dimensional vector}}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \boldsymbol{\theta}^T \cdot \mathbf{x}$$

Vectorized Notation

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \underbrace{2(h_{\boldsymbol{\theta}}(\mathbf{x}) - y)}_{\text{scalar}} \cdot \underbrace{\mathbf{x}}_{(n+1)\text{-dimensional vector}}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \boldsymbol{\theta}^T \cdot \mathbf{x}$$

$$\Downarrow$$

$$\boxed{\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)\mathbf{x}}$$

Vectorized Notation

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)\mathbf{x}$$

Vectorized Notation

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)\mathbf{x}$$

$$= \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_0 \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix} = \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y) \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix}$$

Vectorized Notation

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)\mathbf{x}$$

$$= \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_0 \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix} = \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y) \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix}$$

x_0 = 1 by definition

Vectorized Notation

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)\mathbf{x}$$

$$= \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_0 \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix} = \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y) \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix}$$

x_0 = 1 by definition

The resulting gradient is an $(n+1)$-dimensional vector as expected!

Setting the Gradient Equal to Zero

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y) \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}$$

Setting the Gradient Equal to Zero

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \begin{bmatrix} 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y) \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_1 \\ \vdots \\ 2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}$$

We need to solve a system of *n+1* linear equations with *n+1* variables

$$2(\boldsymbol{\theta}^T \cdot \mathbf{x} - y)x_j = 0 \ \forall j \in \{0, 1, \ldots, n\}$$

Computing the Gradient of MSE (*m* instances)

In the general case where the dataset *D* contains a *m* instances

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\left[\sum_{i=1}^{m}\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)\nabla\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)\right]$$

Computing the Gradient of MSE (*m* instances)

In the general case where the dataset *D* contains a *m* instances

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\left[\sum_{i=1}^{m}\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)\nabla\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)\right]$$

$$\Downarrow$$

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\left[\sum_{i=1}^{m}\underbrace{\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)}_{\text{scalar}}\underbrace{\mathbf{x}_i}_{n+1\text{-dimensional vector}}\right]$$

Computing the Gradient of MSE (*m* instances)

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\left[\sum_{i=1}^{m} \underbrace{\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)}_{\text{scalar}} \underbrace{\mathbf{x}_i}_{n+1\text{-dimensional vector}}\right]$$

Computing the Gradient of MSE (*m* instances)

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\left[\sum_{i=1}^{m} \underbrace{\Big(h_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i\Big)}_{\text{scalar}} \underbrace{\mathbf{x}_i}_{n+1\text{-dimensional vector}}\right]$$

$$\Downarrow$$

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \begin{bmatrix} \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,0} + \ldots + \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,0} \\ \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,1} + \ldots + \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,1} \\ \vdots \\ \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,n} + \ldots + \frac{2}{m}(\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,n} \end{bmatrix}$$

Computing the Gradient of MSE (*m* instances)

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m} \begin{bmatrix} (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,1} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,1} \\ \vdots \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,n} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,n} \end{bmatrix}$$

Computing the Gradient of MSE (*m* instances)

$x_{i,0} = 1$ for all $i = \{1, \ldots, m\}$

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m} \begin{bmatrix} (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,1} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,1} \\ \vdots \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1)x_{1,n} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m)x_{m,n} \end{bmatrix}$$

Computing the Gradient of MSE (*m* instances)

$x_{i,0} = 1$ for all $i = \{1, \ldots, m\}$

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m} \begin{bmatrix} (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) x_{1,1} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) x_{m,1} \\ \vdots \\ (\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) x_{1,n} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) x_{m,n} \end{bmatrix}$$

Again, we need to solve a system of $n+1$ linear equations with $n+1$ variables

$$\frac{2}{m} \left[(\boldsymbol{\theta}^T \cdot \mathbf{x}_1 - y_1) x_{1,j} + \ldots + (\boldsymbol{\theta}^T \cdot \mathbf{x}_m - y_m) x_{m,j} \right] = 0 \ \forall j \in \{0, \ldots, n\}$$

Matrix Notation

$$\mathbf{X} = \underbrace{\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \dots & x_{m,n} \end{bmatrix}}_{m \times n+1 \textbf{ feature matrix}} = \begin{bmatrix} -\mathbf{x}_1^T- \\ -\mathbf{x}_2^T- \\ \vdots \\ -\mathbf{x}_m^T- \end{bmatrix}$$

Matrix Notation

$$\mathbf{X} = \underbrace{\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \dots & x_{m,n} \end{bmatrix}}_{m \times n+1 \textbf{ feature matrix}} = \begin{bmatrix} -\mathbf{x}_1^T- \\ -\mathbf{x}_2^T- \\ \vdots \\ -\mathbf{x}_m^T- \end{bmatrix}$$

$$\boldsymbol{\theta} = \underbrace{\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}}_{n+1\text{-dimensional } \textbf{parameter vector}}$$

Matrix Notation

$$\mathbf{X} = \underbrace{\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \dots & x_{m,n} \end{bmatrix}}_{m \times n+1 \textbf{ feature matrix}} = \begin{bmatrix} -\mathbf{x}_1^T- \\ -\mathbf{x}_2^T- \\ \vdots \\ -\mathbf{x}_m^T- \end{bmatrix}$$

$$\boldsymbol{\theta} = \underbrace{\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}}_{n+1\text{-dimensional } \textbf{parameter vector}} \qquad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{m\text{-dimensional } \textbf{target vector}}$$

Vectorized Form of the Optimization Problem

$$\boxed{h^* = h_{\boldsymbol{\theta}^*} = \operatorname{argmin}_{\boldsymbol{\theta}} \left[\underbrace{\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2}_{\mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D})} \right]}$$

Vectorized Form of the Gradient of MSE

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\mathbf{X}^T(\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y})$$

Vectorized Form of the Gradient of MSE

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m}\mathbf{X}^T(\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y})$$

$$\frac{2}{m}\mathbf{X}^T(\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}) = \mathbf{0}$$

Vectorized Form of the Gradient of MSE

$$\nabla \mathrm{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m} \mathbf{X}^T (\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y})$$

$$\frac{2}{m} \mathbf{X}^T (\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}) = \mathbf{0}$$

$$\mathbf{X}^T \mathbf{X} \cdot \boldsymbol{\theta} = \mathbf{X}^T \cdot \mathbf{y}$$

Vectorized Form of the Gradient of MSE

$$\nabla \text{MSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{2}{m} \mathbf{X}^T (\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y})$$

$$\frac{2}{m} \mathbf{X}^T (\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}) = \mathbf{0}$$

$$\mathbf{X}^T \mathbf{X} \cdot \boldsymbol{\theta} = \mathbf{X}^T \cdot \mathbf{y}$$

$$\boxed{\boldsymbol{\theta} = \mathbf{X}^\dagger \cdot \mathbf{y}}$$

$\mathbf{X}^\dagger = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is the **pseudo-inverse** of $\mathbf{X}$

The Pseudo-Inverse of X

- In general, the feature matrix X is non-squared therefore non-invertible

The Pseudo-Inverse of X

- In general, the feature matrix X is non-squared therefore non-invertible
- $X^T X$ is instead square (n-by-n) and very likely invertible
  - The chance of a randomly generated squared matrix is invertible approaches 1
  - To be non-invertible, the determinant must be 0 (linearly dependent columns)

The Pseudo-Inverse of X

- In general, the feature matrix X is non-squared therefore non-invertible
- $X^T X$ is instead square (*n*-by-*n*) and very likely invertible
  - The chance of a randomly generated squared matrix is invertible approaches 1
  - To be non-invertible, the determinant must be 0 (linearly dependent columns)
- Typically, the number *m* of rows (instances) are way larger than the number *n* of columns (features)
  - $X^T X$ is smaller than X

Additional Notes on OLS

- OLS is also known as one-step learning as there exists a closed-form (i.e., analytical) solution to the convex optimization problem

Additional Notes on OLS

- OLS is also known as one-step learning as there exists a closed-form (i.e., analytical) solution to the convex optimization problem
- However, other choices of loss functions (even if convex) may need an iterative approach to get to a (local) minimum

Additional Notes on OLS

- OLS is also known as one-step learning as there exists a closed-form (i.e., analytical) solution to the convex optimization problem
- However, other choices of loss functions (even if convex) may need an iterative approach to get to a (local) minimum
- Though in general $n << m$, computing the inverse of an n-by-n matrix is still a costly operation ($O(n^3)$ time complexity*)

*$O(n^{2.376})$ using the Coppersmith-Winograd algorithm or similar [ref]

Errors vs. Residuals

Subtle yet important difference between errors and residuals

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

i-th unobservable error

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

i-th unobservable error

i-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$$

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

i-th unobservable error

i-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$$

$$e_i = \hat{y}_i - y_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \underbrace{y_i}_{f(\mathbf{x}_i)+\epsilon_i}$$

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

i-th unobservable error

i-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$$

$$e_i = \hat{y}_i - y_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \underbrace{y_i}_{f(\mathbf{x}_i)+\epsilon_i}$$

i-th residual

Errors vs. Residuals

Subtle yet important difference between errors and residuals

i-th observation

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

i-th unobservable error

i-th prediction

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$$

$$e_i = \hat{y}_i - y_i = h_{\boldsymbol{\theta}}(\mathbf{x}_i) - \underbrace{y_i}_{f(\mathbf{x}_i)+\epsilon_i}$$

i-th residual

MSE is computed from residuals, not unobservable errors!

OLS Assumptions

- Weak exogeneity → Predictor variables (i.e., features) can be treated as error-free constants

OLS Assumptions

- Weak exogeneity → Predictor variables (i.e., features) can be treated as error-free constants
- Linearity → Linear relationship between the features and the response
  - Only a restriction on the parameters; features themselves can be arbitrarily combined using non-linear transformations

OLS Assumptions

- Weak exogeneity → Predictor variables (i.e., features) can be treated as error-free constants
- Linearity → Linear relationship between the features and the response
  - Only a restriction on the parameters; features themselves can be arbitrarily combined using non-linear transformations
- Error independence → Error terms ε_i are uncorrelated with each other
  - Knowing that ε_i is positive (negative) gives no information on the sign of ε_{i+1}

OLS Assumptions

- Homoscedasticity→ Different values of the response variable have the same variance in their errors, regardless of the feature values
  - In practice, this does not hold when the response varies over a wide scale

OLS Assumptions

- Homoscedasticity → Different values of the response variable have the same variance in their errors, regardless of the feature values
  - In practice, this does not hold when the response varies over a wide scale
- No Multicollinearity → There must not be two or more features whose values are perfectly correlated with each other
  - The feature matrix $\mathbf{X}$ must have full column rank n
  - If $\mathbf{X}$ is full column rank n then $\mathbf{X}^T\mathbf{X}$ is always invertible
    - It can be shown that if $\mathbf{X}^T\mathbf{X}\mathbf{u} = \mathbf{0}$ for some vector $\mathbf{u}$, then $\mathbf{u} = \mathbf{0}$ (trivial solution)

Checking OLS Assumptions

- A good way to assess the OLS assumptions hold is to use residual plots

Checking OLS Assumptions

- A good way to assess the OLS assumptions hold is to use residual plots
- Plotting residuals against each feature and/or the predicted value may help spot:
  - Non-linearity
  - Correlation between error terms
  - Non-constant variance of error terms (i.e., heteroscedasticity)
  - …

Assessing the Quality of a Model

- Suppose we have fit a linear regression model to some dataset of observations $D = \{(\mathbf{x}_i, y_i)\}_{i=1..m}$

Assessing the Quality of a Model

- Suppose we have fit a linear regression model to some dataset of observations $D = \{(\mathbf{x}_i, y_i)\}_{i=1..m}$
- In other words, we estimated the vector of parameters $\boldsymbol{\theta}^*$ using OLS

Assessing the Quality of a Model

- Suppose we have fit a linear regression model to some dataset of observations $D = \{(\mathbf{x}_i, y_i)\}_{i=1..m}$
- In other words, we estimated the vector of parameters $\boldsymbol{\theta}^*$ using OLS

How do we measure the "goodness-of-fit" of the model?

Assessing the Quality of a Model

- Suppose we have fit a linear regression model to some dataset of observations $D = \{(\mathbf{x}_i, y_i)\}_{i=1..m}$
- In other words, we estimated the vector of parameters $\boldsymbol{\theta}^*$ using OLS

How do we measure the "goodness-of-fit" of the model?

Residual Standard Error (RSE)

Assessing the Quality of a Model

- Suppose we have fit a linear regression model to some dataset of observations $D = \{(x_i, y_i)\}_{i=1..m}$
- In other words, we estimated the vector of parameters $\boldsymbol{\theta}^*$ using OLS

How do we measure the "goodness-of-fit" of the model?

Residual Standard Error (RSE)

R^2 statistic

Residual Standard Error (RSE)

Recall that every observation of the target variable y_i is associated with an error term ε_i

$$y_i = \underbrace{\theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}}_{h_{\boldsymbol{\theta}}(\mathbf{x}_i) \;\approx\; f(\mathbf{x}_i)} + \epsilon_i$$

Residual Standard Error (RSE)

Recall that every observation of the target variable y_i is associated with an error term ε_i

$$y_i = \underbrace{\theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}}_{h_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx f(\mathbf{x}_i)} + \boxed{\epsilon_i}$$

Residual Standard Error (RSE)

Recall that every observation of the target variable y_i is associated with an error term ε_i

$$y_i = \underbrace{\theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n}}_{h_{\boldsymbol{\theta}}(\mathbf{x}_i) \, \approx \, f(\mathbf{x}_i)} + \boxed{\epsilon_i}$$

Even if we were able to find the exact parameters of the true f, we would not be able to perfectly predict y_i from $\mathbf{x}_i$

Residual Standard Error (RSE)

$\boxed{\text{RSE is an estimate of the standard deviation of } \varepsilon}$

$$\mathrm{RSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \sqrt{\frac{1}{\underbrace{m-n-1}_{\textbf{degrees of freedom}}} \underbrace{\sum_{i=1}^{m} (\hat{y}_i - y_i)^2}_{\mathrm{RSS}}}$$

Residual Standard Error (RSE)

RSE is an estimate of the standard deviation of $\boldsymbol{\varepsilon}$

$$\mathrm{RSE}(h_{\boldsymbol{\theta}}, \mathcal{D}) = \sqrt{\frac{1}{\underbrace{m-n-1}_{\textbf{degrees of freedom}}} \underbrace{\sum_{i=1}^{m} (\hat{y}_i - y_i)^2}_{\mathrm{RSS}}}$$

A measure of the lack of fit of the model to the data
the lower the better

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$

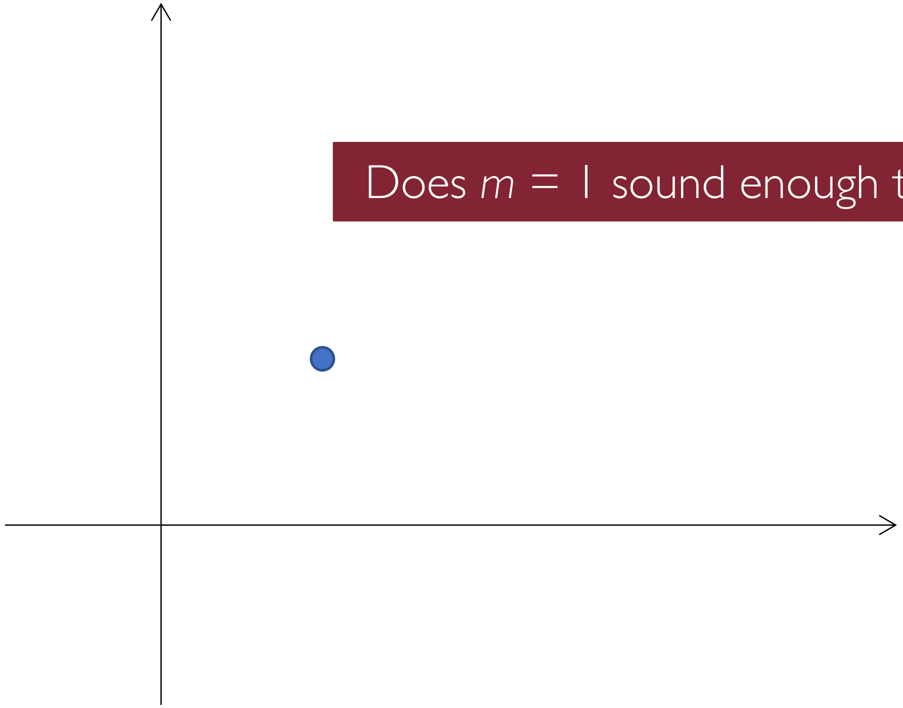
How many observations *m* do I need to estimate model's parameters?

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$

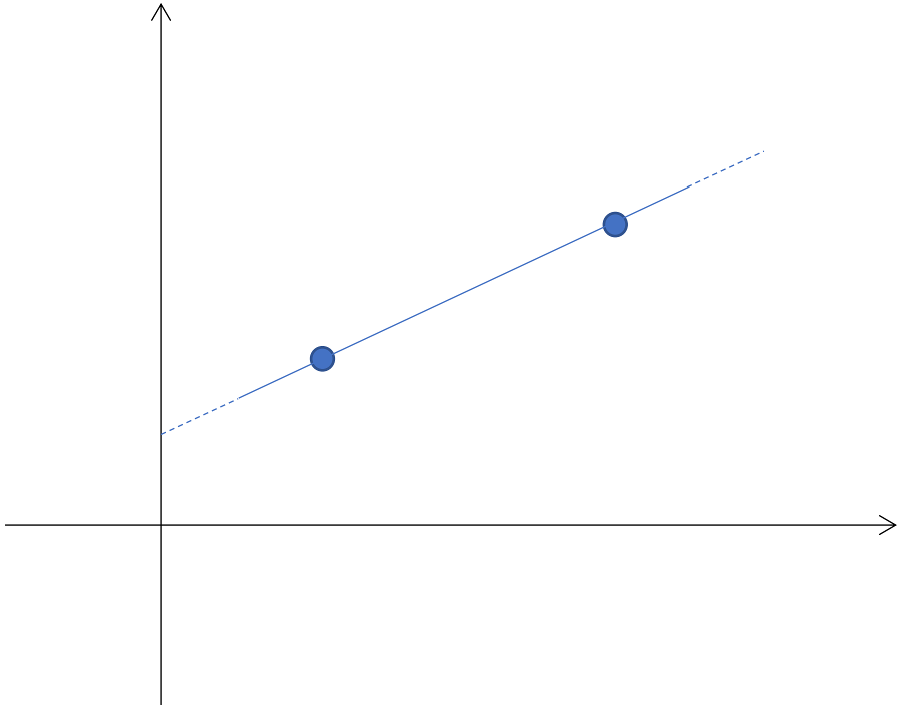
How many observations m do I need to estimate model's parameters?

Does m = 1 sound enough to fit a line?



Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$

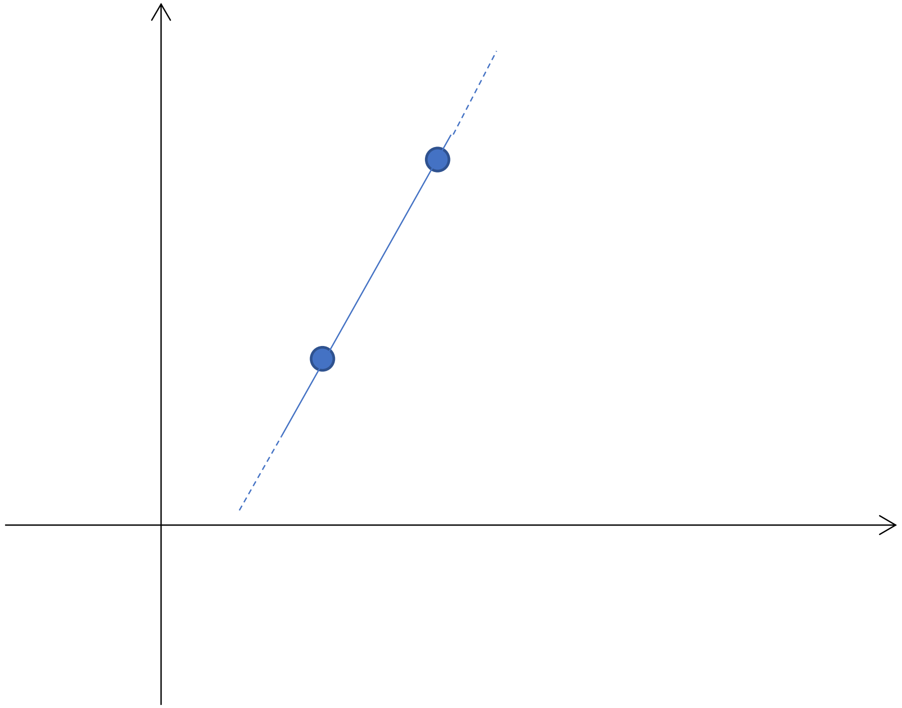


How many observations m do I need to estimate model's parameters?

With 2 data points I am always able to fit a perfect line

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$



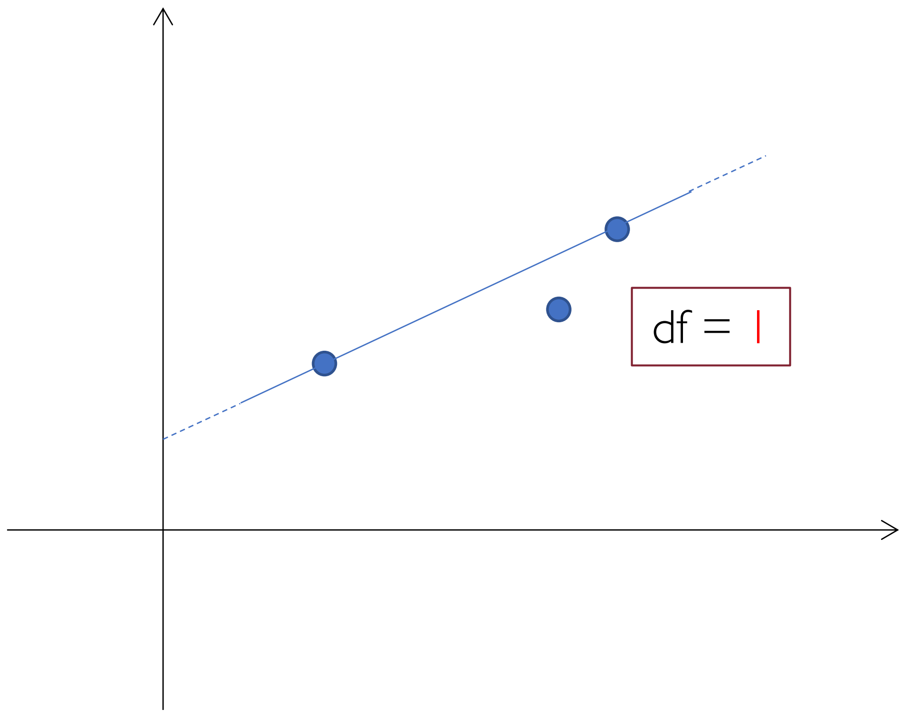
How many observations *m* do I need to estimate model's parameters?

With 2 data points I am always able to fit a perfect line

Problem is that my fitted line may drastically change depending on where the second point is located!

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$



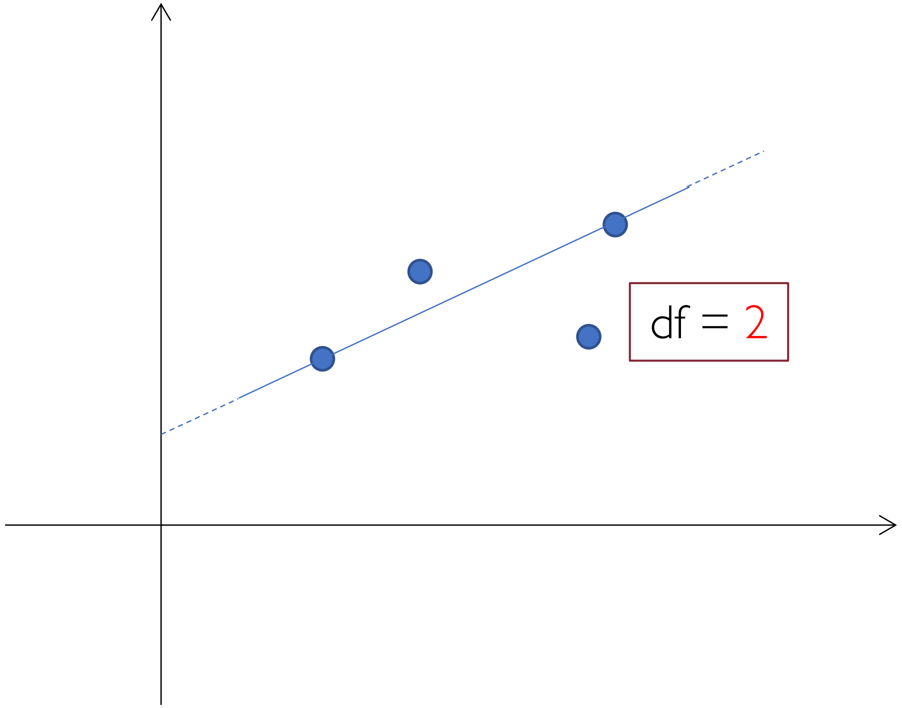
How many observations *m* do I need to estimate model's parameters?

With 2 data points I am always able to fit a perfect line

If I want my model to be more "flexible" I need **at least** 3 points which leave me with 1 degree of freedom

Degrees of Freedom

$$y_i = \theta_0 + \theta_1 x_{i,1} + \epsilon_i$$



How many observations *m* do I need to estimate model's parameters?

With 2 data points I am always able to fit a perfect line

If I want my model to be more "flexible" I need **at least** 3 points which leave me with 1 degree of freedom

Degrees of Freedom

What happens when we add more variables to the model?

$$y_i = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n} + \epsilon_i$$

Degrees of Freedom

What happens when we add more variables to the model?

$$y_i = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n} + \epsilon_i$$

Given the same number of observations *m*,
we loose 1 degree of freedom for each variable we add

Degrees of Freedom

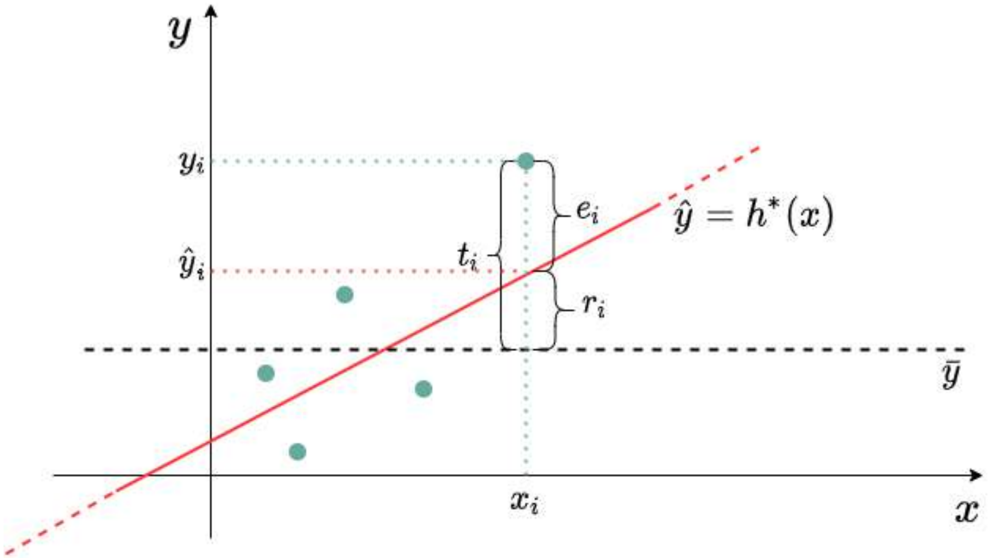
What happens when we add more variables to the model?

$$y_i = \theta_0 x_{i,0} + \theta_1 x_{i,1} + \ldots + \theta_n x_{i,n} + \epsilon_i$$

Given the same number of observations *m*,
we loose 1 degree of freedom for each variable we add

$$\boxed{\text{df} = \underbrace{m}_{\#\text{observations}} - \underbrace{n}_{\#\text{features}} - \underbrace{1}_{\text{intercept}}}$$

R^2 Statistic



$$t_i = y_i - \bar{y}$$
$$e_i = y_i - \hat{y}_i$$
$$r_i = \hat{y}_i - \bar{y}$$

$$TSS = \sum_{i=1}^{m} (y_i - \bar{y})^2 = \sum_{i=1}^{m} t_i^2$$

$$RSS = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} e_i^2$$

$$R^2 = 1 - \frac{RSS}{TSS} = \frac{TSS - RSS}{TSS}$$

R^2 Statistic

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=1}^{m} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{m} (y_i - \bar{y})^2}$$

TSS measures the total variance in the response Y **before** the regression takes place

R^2 Statistic

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

TSS measures the total variance in the response Y **before** the regression takes place

RSS measures the amount of variability that is **left unexplained** after performing the regression

R^2 Statistic

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

TSS measures the total variance in the response Y **before** the regression takes place

RSS measures the amount of variability that is **left unexplained** after performing the regression

R^2 measures the proportion of variability in Y that can be explained using X

R^2 Statistic

- An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression

R^2 Statistic

- An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression
- The larger R^2 the better is the linear regression model

R^2 Statistic

- An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression
- The larger R^2 the better is the linear regression model
- R^2 is easier to interpret than RSE as it always ranges between 0 and 1

Adjusted R^2 Statistic

- Fixing the sample size m, RSS decreases (or, at worst, it stays the same) as more variables are added to the fitted model

Adjusted R^2 Statistic

- Fixing the sample size m, RSS decreases (or, at worst, it stays the same) as more variables are added to the fitted model
- R^2 always increases as more variables are added (as df decreases!)

Adjusted R^2 Statistic

- Fixing the sample size m, RSS decreases (or, at worst, it stays the same) as more variables are added to the fitted model
- R^2 always increases as more variables are added (as df decreases!)
- We need a way to adjust for that, otherwise we could get a better model by simply adding useless features to it!

$$R^2_{\text{adj}} = 1 - \frac{\frac{\text{RSS}}{m-n-1}}{\frac{\text{TSS}}{m-1}}$$

Adjusted R^2 Statistic

- Maximizing the adjusted R^2 is equivalent to minimizing RSS/(m-n-1)

Adjusted R^2 Statistic

- Maximizing the adjusted R^2 is equivalent to minimizing RSS/(m-n-1)
- We know RSS may decrease if the number of variables in the model increases

Adjusted R^2 Statistic

- Maximizing the adjusted R^2 is equivalent to minimizing RSS/(m-n-1)
- We know RSS may decrease if the number of variables in the model increases
- RSS/(m-n-1) may increase or decrease, due to the presence of n in the denominator

Adjusted R^2 Statistic

- Maximizing the adjusted R^2 is equivalent to minimizing RSS/(m-n-1)
- We know RSS may decrease if the number of variables in the model increases
- RSS/(m-n-1) may increase or decrease, due to the presence of n in the denominator
- We may need to increase the sample size m to compensate for the increasing of RSS due to the inclusion of more features n

Regularization

- The absolute value of learned parameters $\boldsymbol{\theta}$ should not be very large
  - Otherwise, a small change in an input feature may cause a high difference in the ouput predicted value

Regularization

- The absolute value of learned parameters $\boldsymbol{\theta}$ should not be very large
  - Otherwise, a small change in an input feature may cause a high difference in the ouput predicted value
- This is an indication of overfitting:
  - The learned model is highly "training set dependent" and does not generalize

Regularization

- The absolute value of learned parameters $\boldsymbol{\theta}$ should not be very large
  - Otherwise, a small change in an input feature may cause a high difference in the ouput predicted value
- This is an indication of overfitting:
  - The learned model is highly "training set dependent" and does not generalize
- Regularization → Put some constraint on the optimization problem so as to limit the values of the learned parameters

Elastic Net Framework

We consider a far more general optimization framework, which OLS is just a special case of

Elastic Net Framework

We consider a far more general optimization framework,
which OLS is just a special case of

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2 + \lambda \Big(\alpha |\boldsymbol{\theta}| + (1-\alpha) ||\boldsymbol{\theta}||^2 \Big) \right]$$

Elastic Net Framework

We consider a far more general optimization framework, which OLS is just a special case of

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2 + \lambda \Big(\alpha |\boldsymbol{\theta}| + (1 - \alpha) ||\boldsymbol{\theta}||^2 \Big) \right]$$

$\lambda \geq 0$ regularization parameter: when this is 0 we backup to OLS (no regularization at all)

Elastic Net Framework

We consider a far more general optimization framework,
which OLS is just a special case of

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2 + \lambda \Big(\alpha |\boldsymbol{\theta}| + (1 - \alpha) ||\boldsymbol{\theta}||^2 \Big) \right]$$

$\lambda \geq 0$ regularization parameter: when this is 0 we backup to OLS (no regularization at all)

$\alpha \in [0, 1]$ tradeoff parameter: to weight regulatization penalties

Elastic Net Framework

We consider a far more general optimization framework,
which OLS is just a special case of

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2 + \lambda \Big(\alpha |\boldsymbol{\theta}| + (1 - \alpha) ||\boldsymbol{\theta}||^2 \Big) \right]$$

$\lambda \geq 0$ regularization parameter: when this is 0 we backup to OLS (no regularization at all)

$\alpha \in [0, 1]$ tradeoff parameter: to weight regulatization penalties

$\lambda > 0; \ \alpha = 1$ Least Absolute Shrinkage and Selection Operator or LASSO (L1-regularization only)

Elastic Net Framework

We consider a far more general optimization framework,
which OLS is just a special case of

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{m} ||\mathbf{X} \cdot \boldsymbol{\theta} - \mathbf{y}||^2 + \lambda \Big(\alpha |\boldsymbol{\theta}| + (1 - \alpha) ||\boldsymbol{\theta}||^2 \Big) \right]$$

$\lambda \geq 0$ regularization parameter: when this is 0 we backup to OLS (no regularization at all)

$\alpha \in [0, 1]$ tradeoff parameter: to weight regulatization penalties

$\lambda > 0; \ \alpha = 1$ Least Absolute Shrinkage and Selection Operator or LASSO (L1-regularization only)

$\lambda > 0; \ \alpha = 0$ Ridge (L2-regularization only)

Take-Home Message of Today

- Linear Regression is a simple yet powerful tool for learning real-valued functions between feature and response variables

Take-Home Message of Today

- Linear Regression is a simple yet powerful tool for learning real-valued functions between feature and response variables
- The estimation of model's parameters is usually done via Ordinary Least Squares (OLS) by minimizing Mean Squared Error (MSE)

Take-Home Message of Today

- Linear Regression is a simple yet powerful tool for learning real-valued functions between feature and response variables
- The estimation of model's parameters is usually done via Ordinary Least Squares (OLS) by minimizing Mean Squared Error (MSE)
- OLS admits a closed-form solution which allows computing the parameters analytically via the pseudo-inverse of the feature matrix $\mathbf{X}$

Take-Home Message of Today

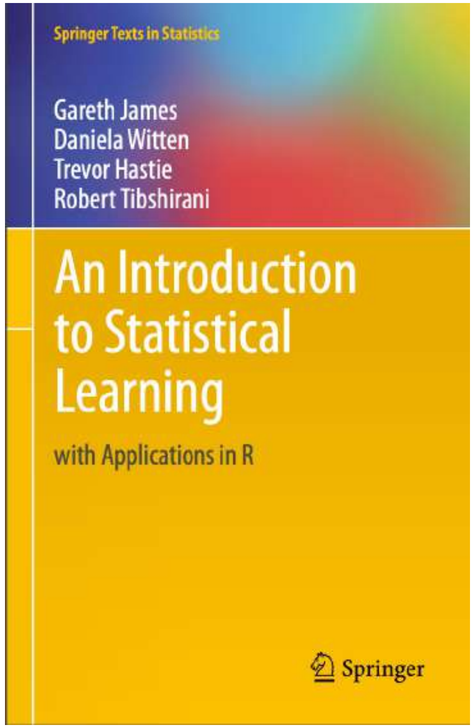
- Linear Regression is a simple yet powerful tool for learning real-valued functions between feature and response variables
- The estimation of model's parameters is usually done via Ordinary Least Squares (OLS) by minimizing Mean Squared Error (MSE)
- OLS admits a closed-form solution which allows computing the parameters analytically via the pseudo-inverse of the feature matrix $\mathbf{X}$
- Several quality measures: RSE, R^2, Adjusted R^2, etc.

Take-Home Message of Today

- Linear Regression is a simple yet powerful tool for learning real-valued functions between feature and response variables
- The estimation of model's parameters is usually done via Ordinary Least Squares (OLS) by minimizing Mean Squared Error (MSE)
- OLS admits a closed-form solution which allows computing the parameters analytically via the pseudo-inverse of the feature matrix $\mathbf{X}$
- Several quality measures: RSE, R^2, Adjusted R^2, etc.
- Regularization to prevent overfitting: Elastic Net, LASSO, Ridge

Further Readings

An Introduction to Statistical Learning [Chapter 3]



Freely available at:
https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf