2
Linear Regression and Gradient Descent

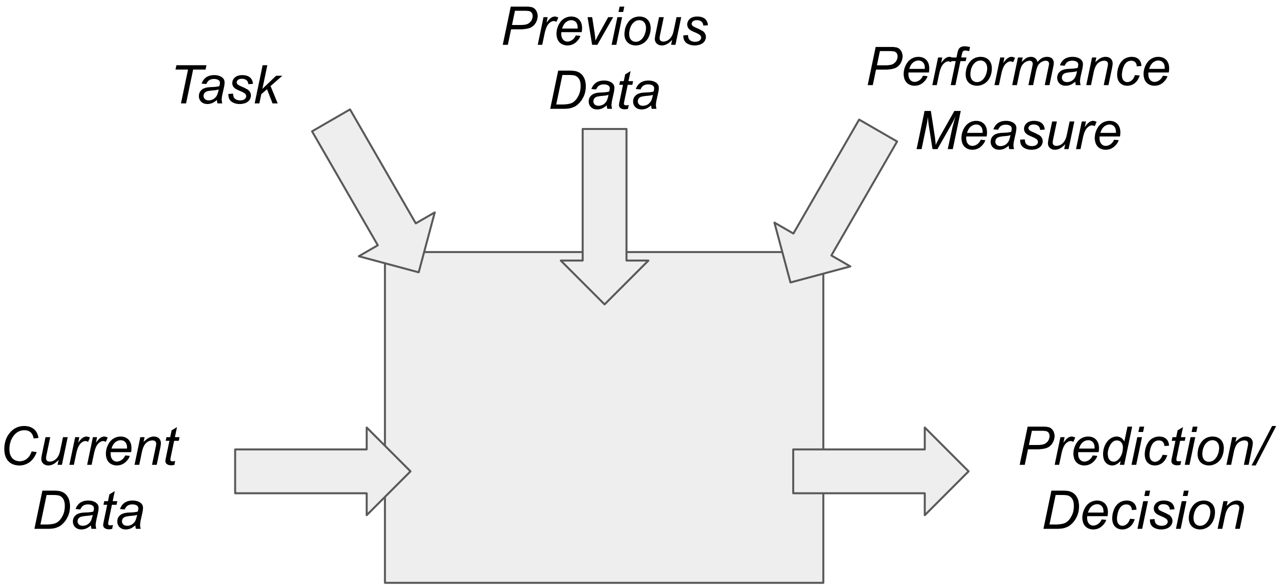
Dr. Amal Aboulhassan

ALEXANDRIA UNIVERSITY

Machine Learning Taxonomy

- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Others
- Reinforcement Learning

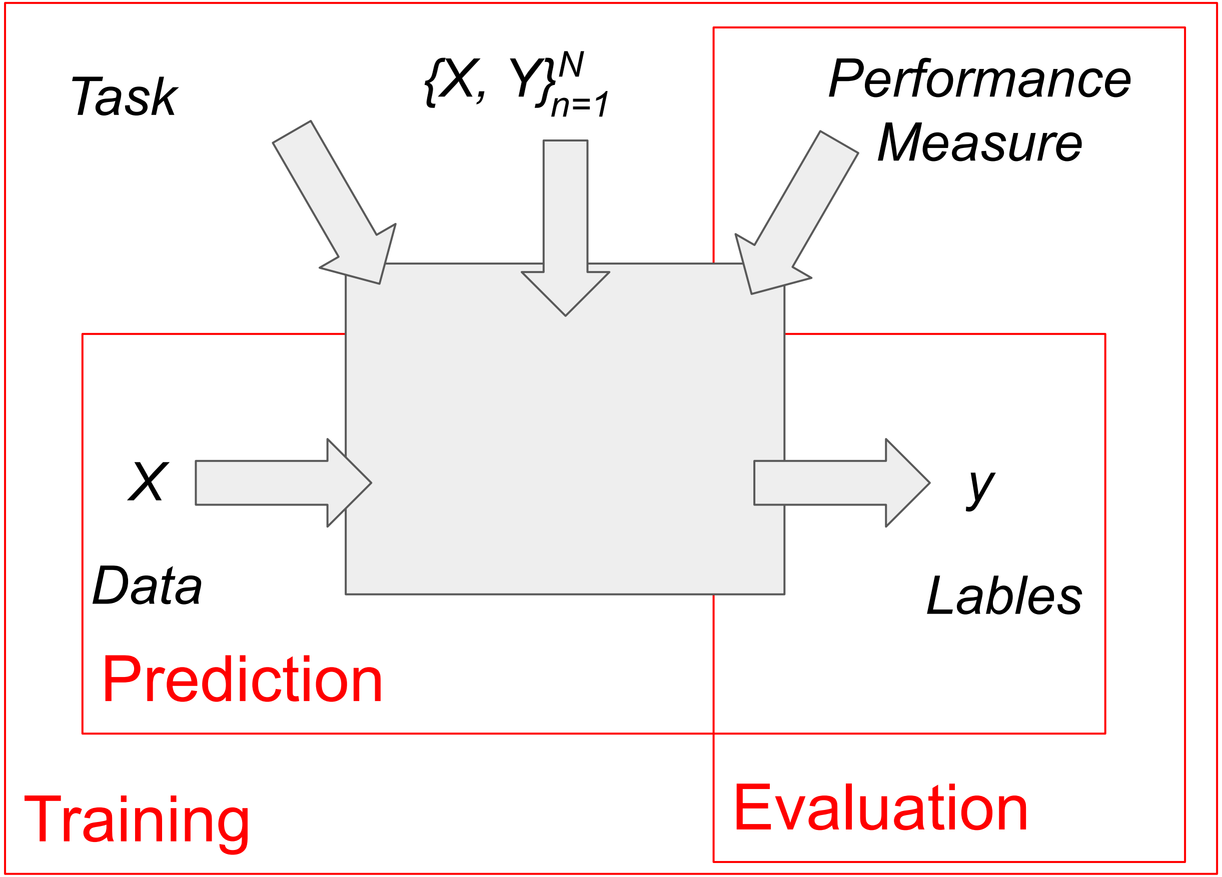
Machine Learning Process



Machine Learning Skills

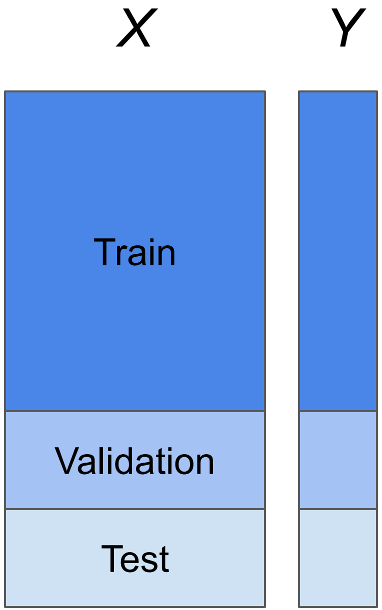
- Data
  - Nature
  - Pre-processing
  - Goals
- Math
  - Algorithm choice
  - Parameters setting
- Programming
  - Python
  - State of the art tools (e.g. SciLearn, Pandas, Matlab, etc.)

Supervised Learning Process

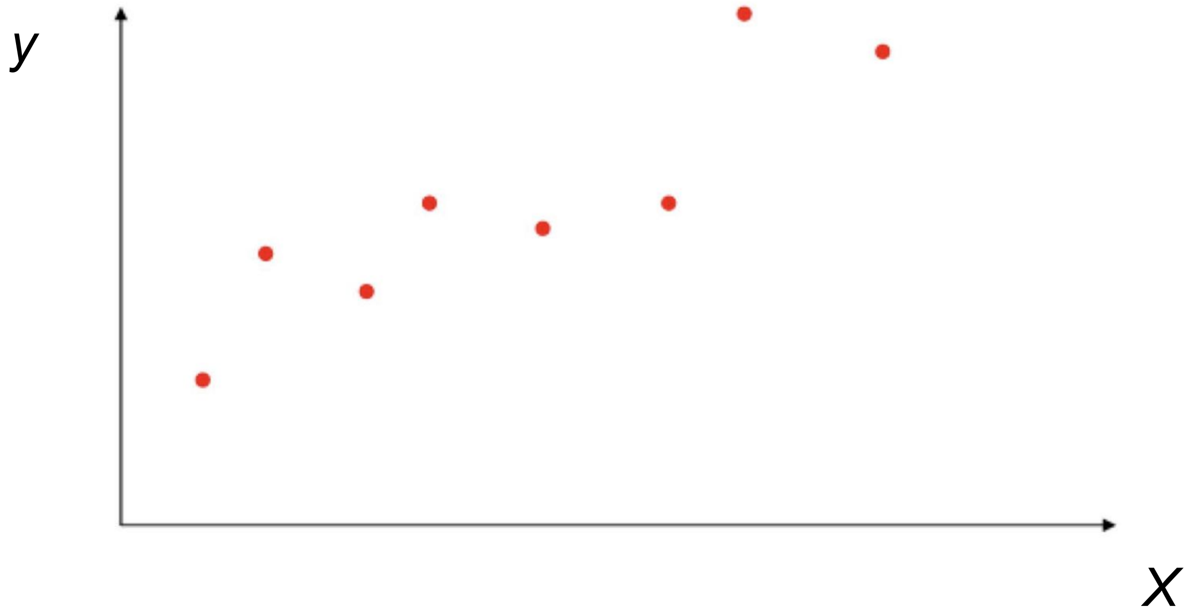


Regression Tasks/Steps

- Training
- Prediction
- Evaluation
    - Choose metrics
    - Different ways of choosing validation data - Data Splitting

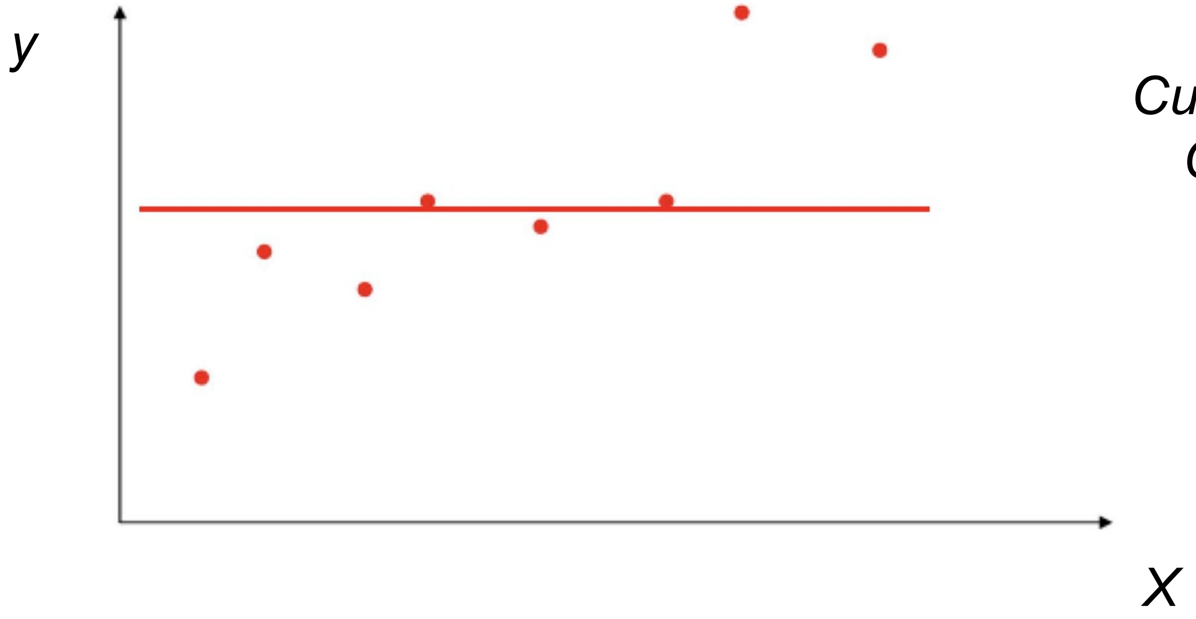


Example



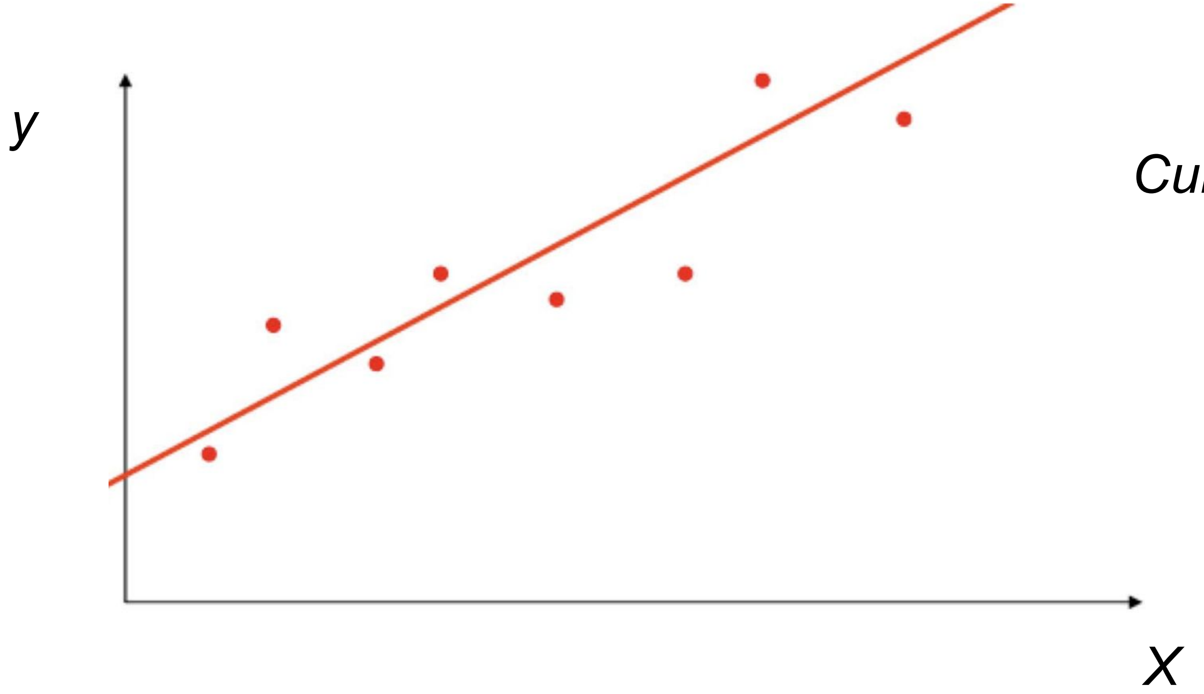
Data

Example



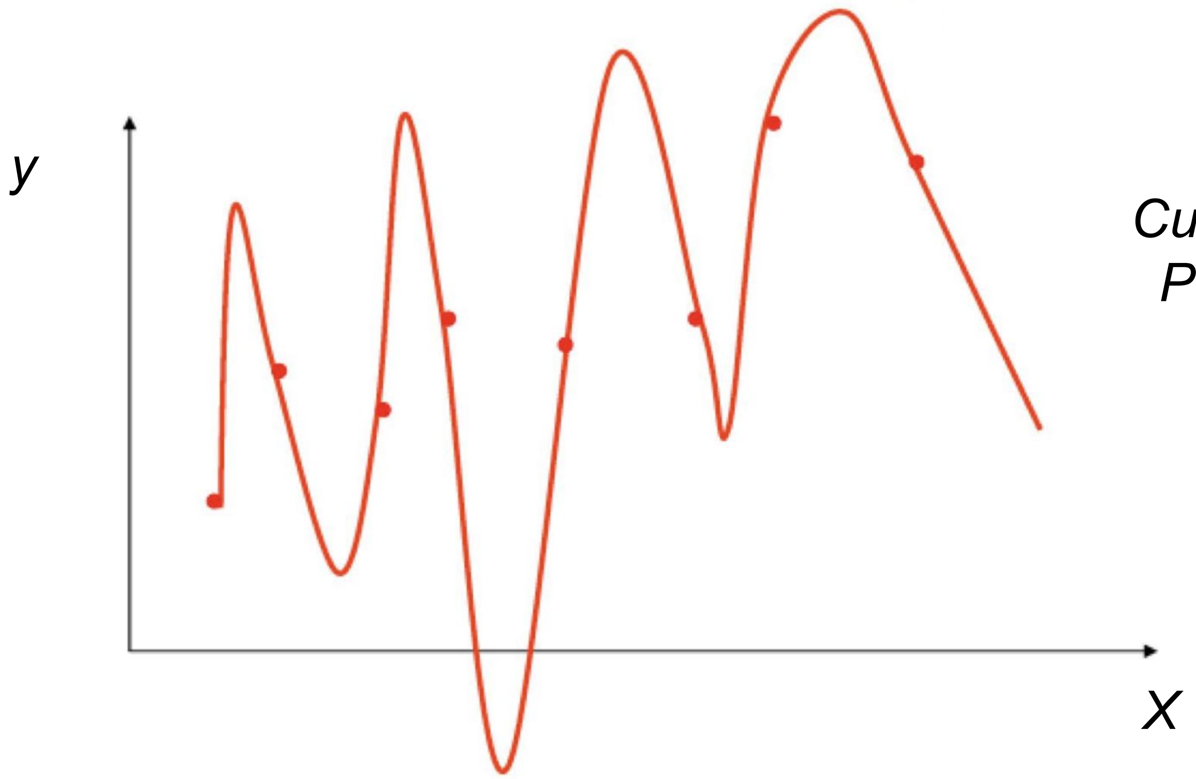
Curve Fitting: Constant

Example



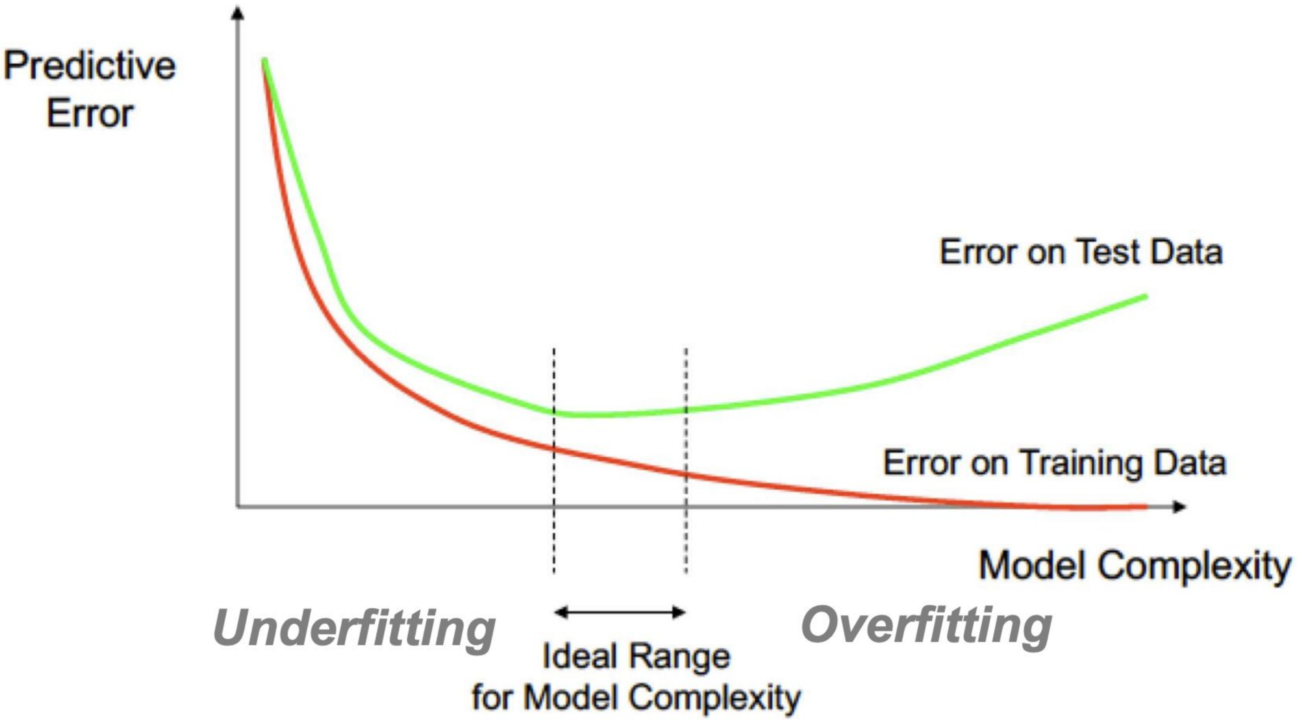
Curve Fitting:
Line

Example



Curve Fitting: Polynomial

Impact of Model Complexity



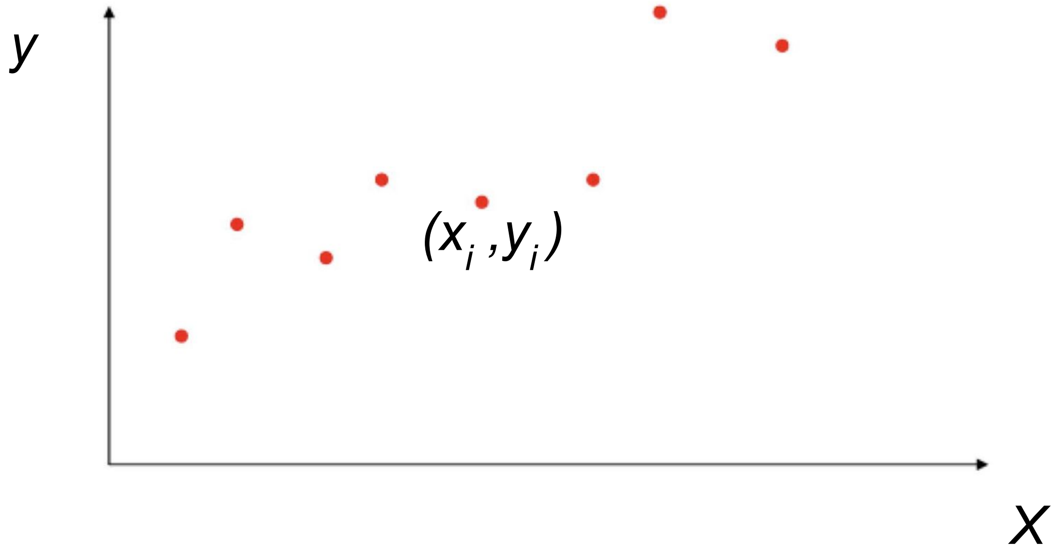
Linear Regression

- Linear regression has been around since more than 200 years.
- Linear regression is a linear model :y can be calculated from a linear combination of the input variables (x).
- When there is:
  - single input variable (x), the method is referred to as simple linear regression.
  - multiple input variables, the method is referred to as multiple linear regression.
- Different techniques can be used to prepare or train the linear regression equation from data:
  - Ordinary Least Squares (or Linear Regression or just Least Squares Regression).
  - Gradient Descent
  - Regularization

Linear Regression

- Training "least squares" linear regression
  - 1-dim. features without intercept
  - 1-dim. features with intercept
  - General case: Many features with intercept
  - Note: bias is another name for intercept

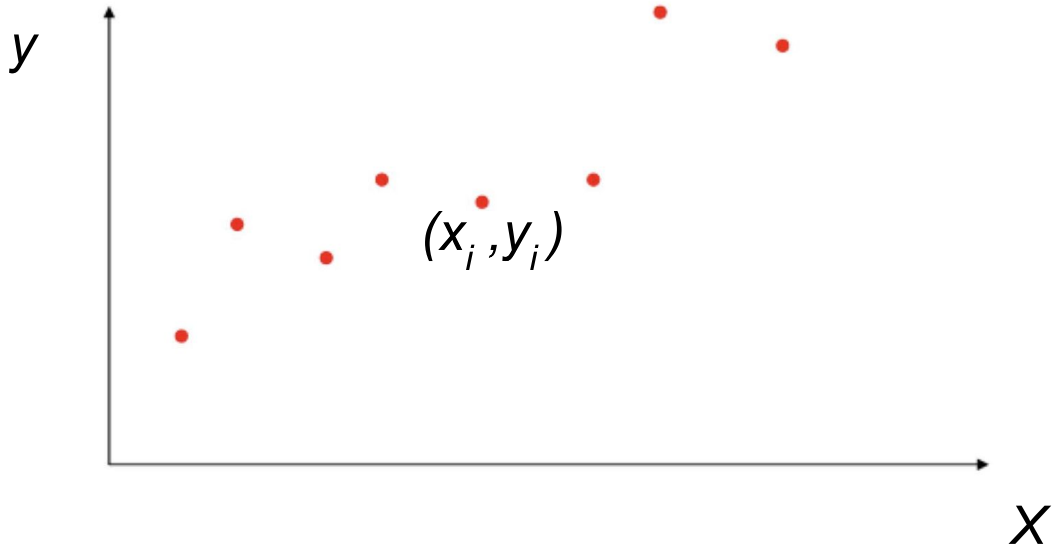
(1) Least Squares: 1 Feature



$$\underset{\text{Dependent Variable}}{Y_i} = \overset{\text{Constant/Intercept}}{b} + \underset{\text{Slope/Coefficient}}{w}\,\overset{\text{Independent Variable}}{X_i}$$

$$\{X, Y\}_{i=1}^{N}$$

(1) Least Squares: 1 Feature

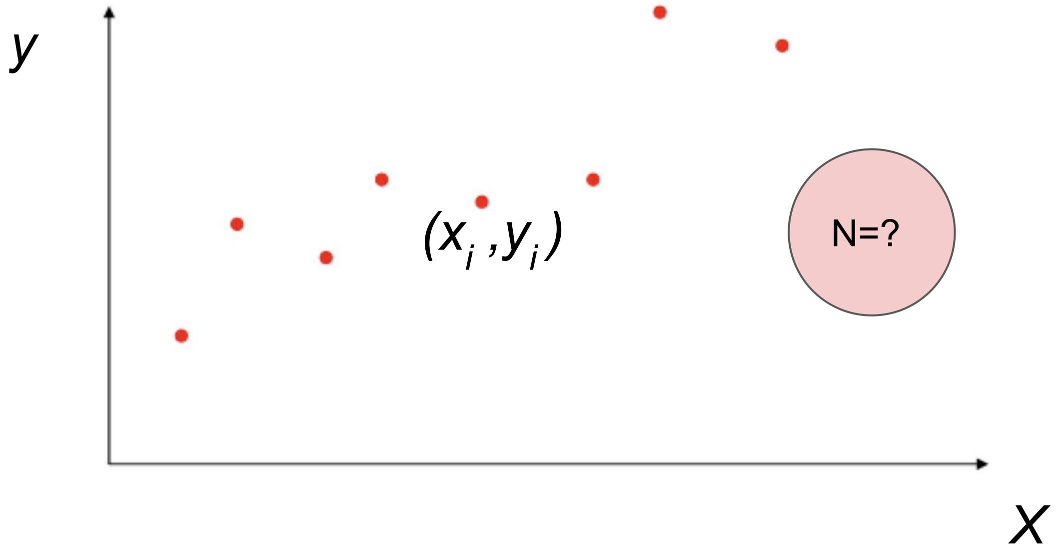


Cons

$$\underbrace{Y_i}_{\text{Prediction}} = \overbrace{b}^{\text{Bias}} + \underbrace{w}_{\text{Weight}} \overbrace{X_i}^{\text{Feature}}$$

$$\{X, Y\}_{i=1}^{N}$$

Linear Regression: (1) Least Squares



Cons

Bias

Feature

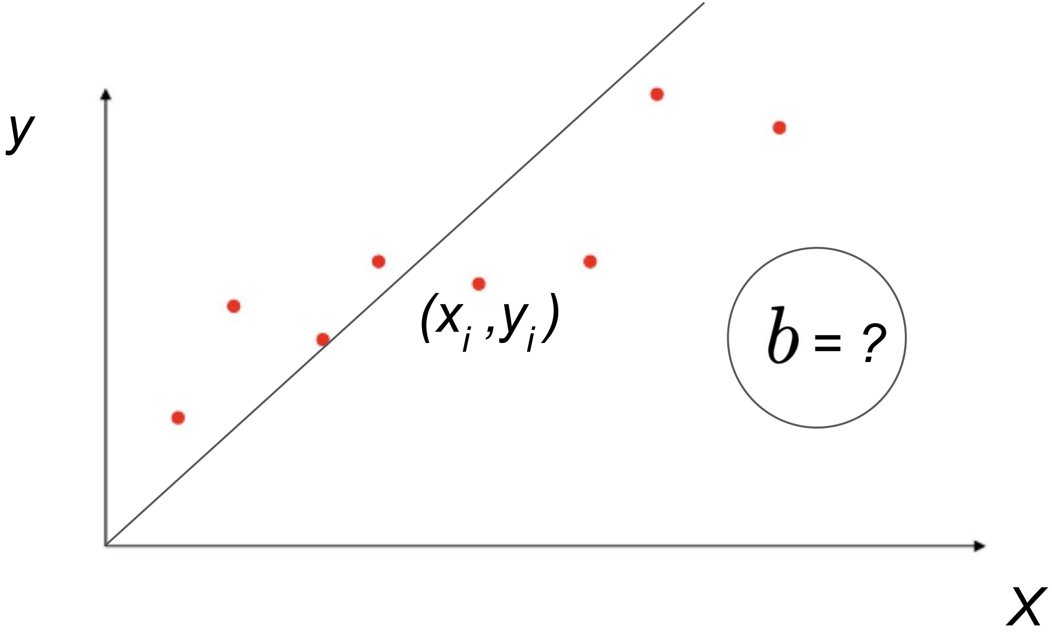
$$Y_i = b + w X_i$$

Prediction

Weight

$$\{X, Y\}_{i=1}^{N}$$

(1) Least Squares: 1 Feature



Cons

Bias

Feature

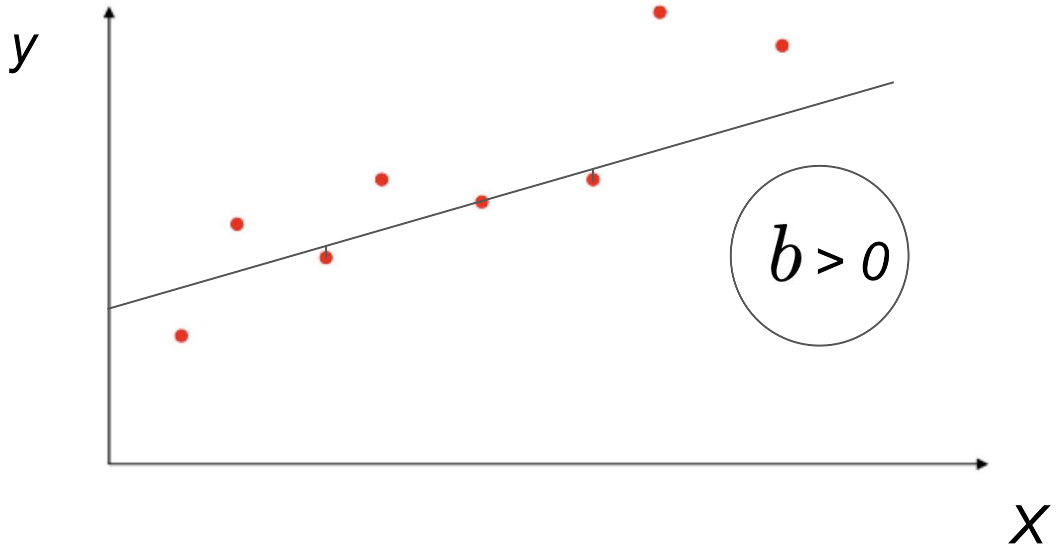
$$Y_i = b + w X_i$$

Prediction

Weight

$\{X, Y\}_{n=1}^{N}$

Linear Regression: (1) Least Squares

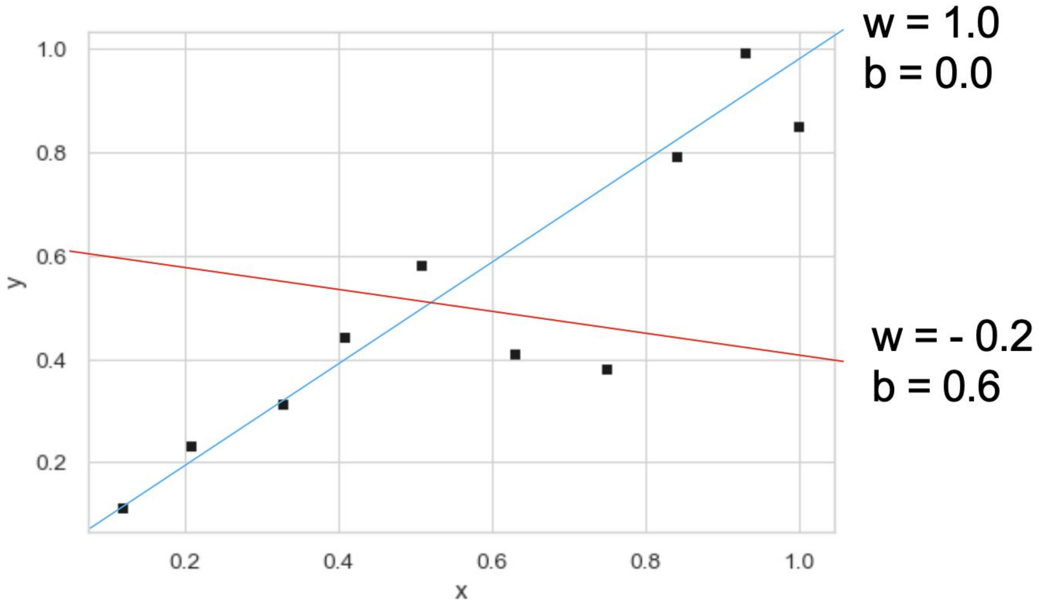


Cons

$$\underbrace{Y_i}_{\text{Prediction}} = \overbrace{b}^{\text{Bias}} + \underbrace{w}_{\text{Weight}} \overbrace{X_i}^{\text{Feature}}$$

$$\{X, Y\}_{n=1}^{N}$$

Linear Regression: (1) Least Squares



Cons

Bias

Feature

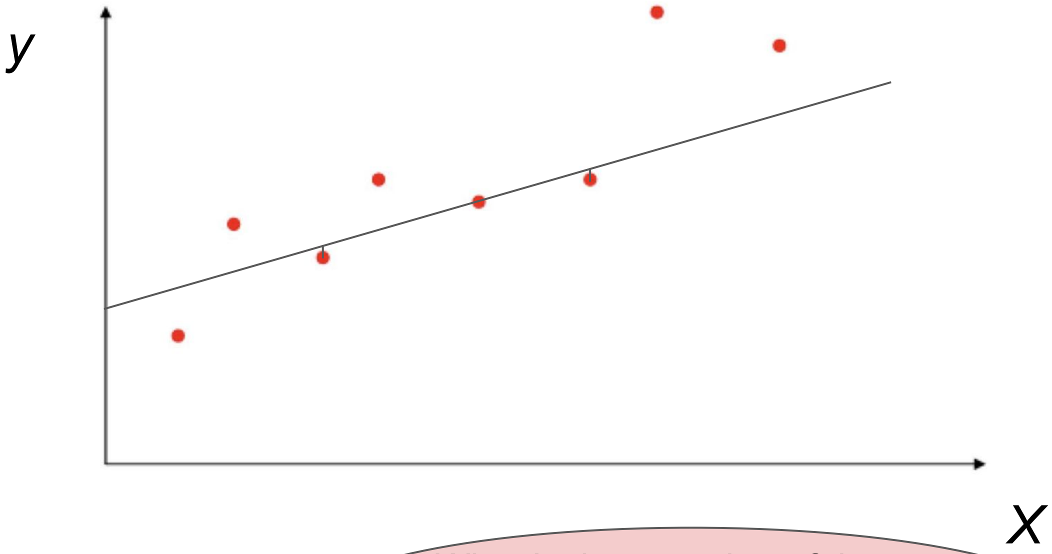
$$Y_i = b + w X_i$$

Prediction

Weight

$$\{X, Y\}_{i=1}^{N}$$

Linear Regression: (1) Least Squares

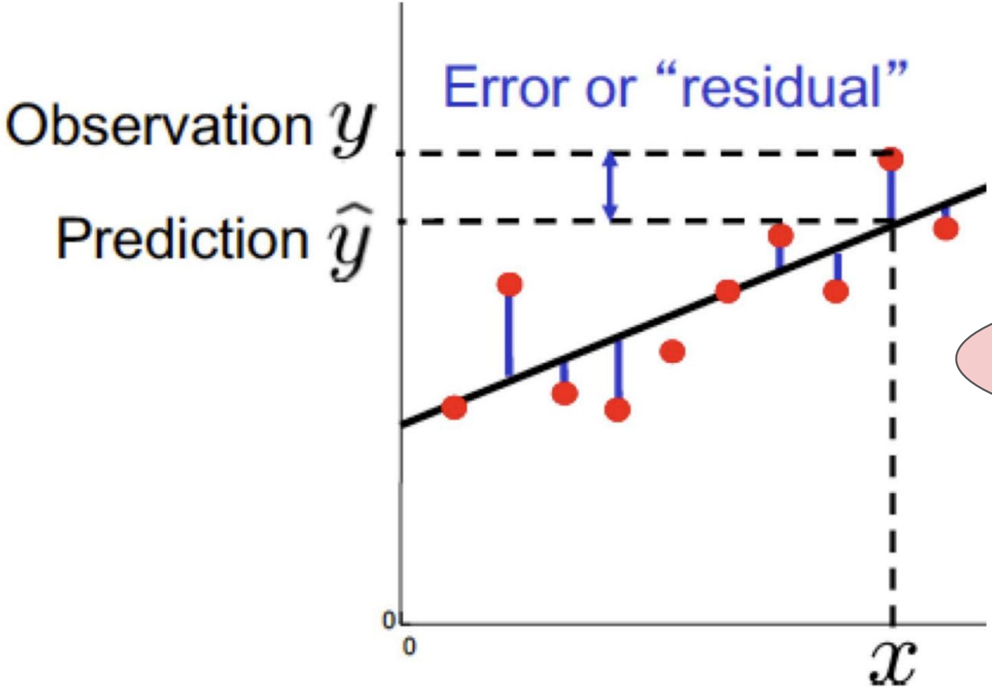


$$\underbrace{Y_i}_{\text{Prediction}} = \overbrace{b}^{\text{Bias}} + \underbrace{w}_{\text{Weight}} \overbrace{X_i}^{\text{Feature}}$$

$\{X, Y\}_{n=1}^{N}$

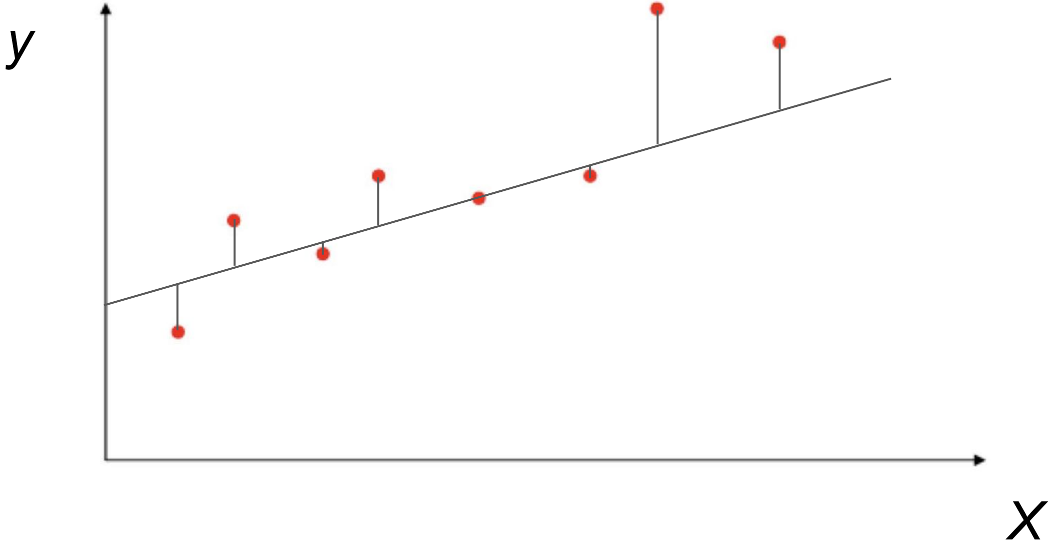
What is the equation of the "Best" fitting line using Least Squares?

Linear Regression: (1) Least Squares



What is the equation of the “Best” fitting line using Least Squares?

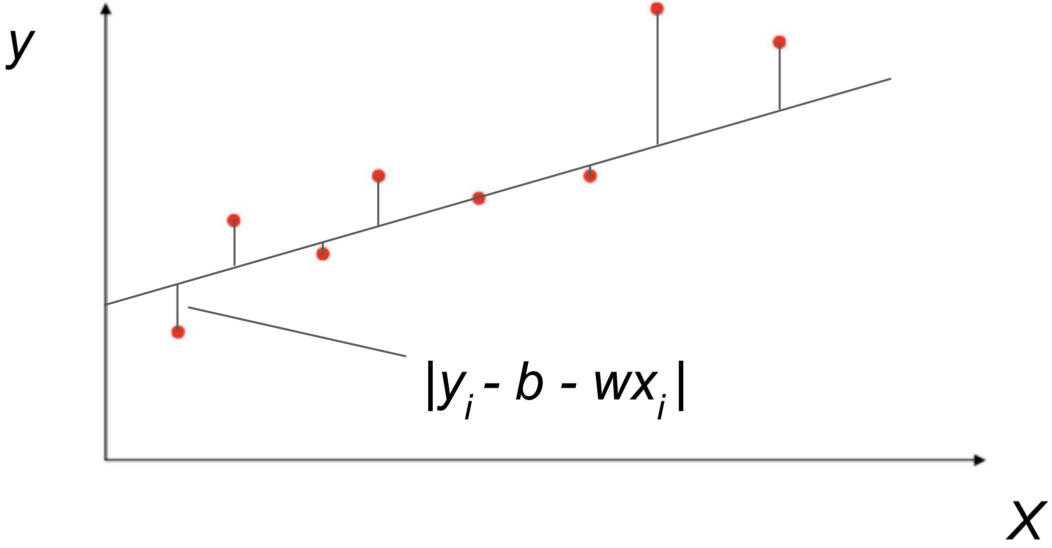
Linear Regression: (1) Least Squares



$$\underbrace{Y_i}_{\text{Prediction}} = \overbrace{b}^{\text{Bias}} + \underbrace{w}_{\text{Weight}}\, \overbrace{X_i}^{\text{Feature}}$$

$$\{X, Y\}_{i=1}^{N}$$

Linear Regression: (1) Least Squares

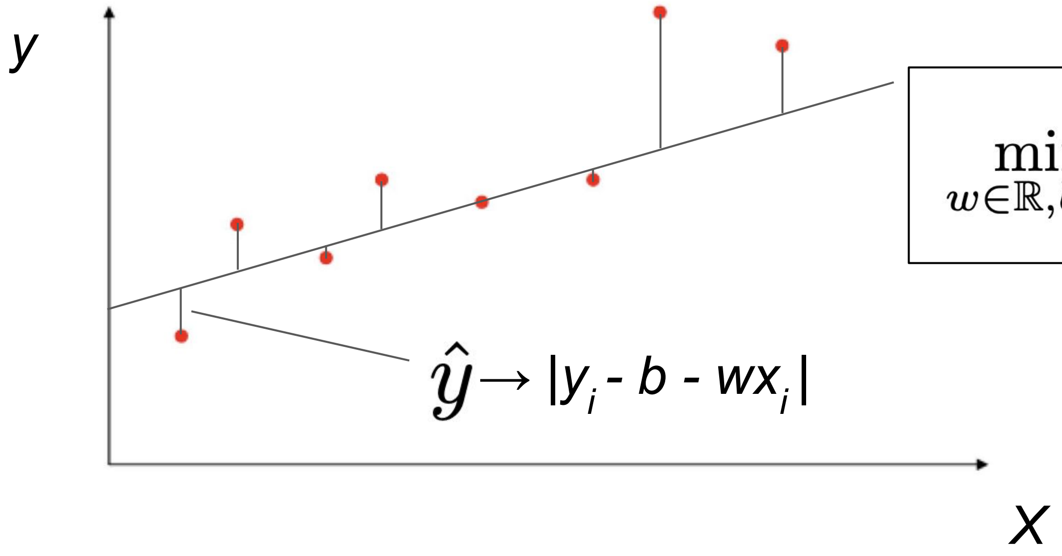


Cons

$$\underset{\text{Prediction}}{Y_i} = \overset{\text{Bias}}{b} + \underset{\text{Weight}}{w}\,\overset{\text{Feature}}{X_i}$$

$$\{X, Y\}_{i=1}^{N}$$

Linear Regression: (1) Least Squares



$$\min_{w \in \mathbb{R}, b \in \mathbb{R}} \sum_{i=1}^{N} (y_i - \hat{y}(x_i, w, b))^2$$

Minimize Cost Function

Linear Regression: (1) Least Squares

- Task: Training
- Training Data: $\{X, Y\}_{n=1}^{N}$
  - X: Features
  - Y: Prediction/Labels/Response
- Model Function: Straight line
- Cost Function: Sum of Squared Errors
- Error:
  - distance between two points observation y and prediction $\hat{y}$
- Learning Algorithm: Linear Least Square
  - Output Model: values of w and b which minimize the cost function on the training set

Linear Regression: (1) Least Squares

- Solution 1: Closed Form/Analytical/Mathematical
- Derivation steps:

1. Compute gradient of objective wrt w, as a function of w and b

2. Compute gradient of objective wrt b, as a function of w and b

3. Set (1) and (2) equal to zero and solve for w and b (2 equations, 2 unknowns)

$$w = \frac{\sum_{n=1}^{N} (x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N} (x_n - \bar{x})^2}$$

$$b = \bar{y} - w\bar{x}$$

$$\bar{x} = \text{mean}(x_1, \ldots x_N)$$
$$\bar{y} = \text{mean}(y_1, \ldots y_N)$$

Linear Regression: (1) Least Squares

- Note: we can write this formula as mean squared errors or distance based on the derivation and the objective of the computation

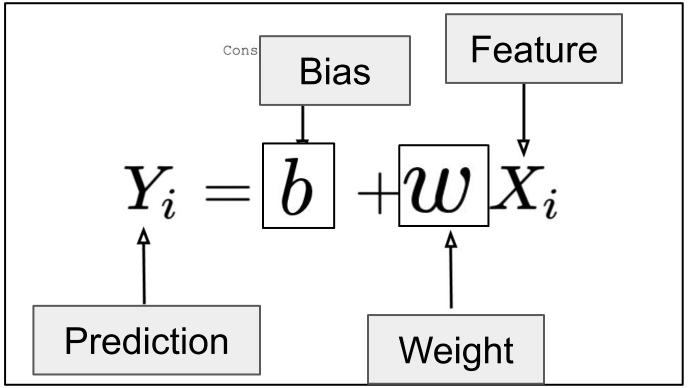
$$w = \frac{\sum_{n=1}^{N}(x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N}(x_n - \bar{x})^2}$$

$$b = \bar{y} - w\bar{x}$$

$$\bar{x} = \text{mean}(x_1, \ldots x_N)$$

$$\bar{y} = \text{mean}(y_1, \ldots y_N)$$

(1) Least Squares: F-dim Features



$\{X, Y\}_{i=1}^{N}$

1-dim Feature

$Y = b + wX$

F-dim Feature

$Y = w_0 + w_1X_1 + w_2X_2 + \ldots w_f x_f$

(1) Least Squares: F-dim Features

$$\underbrace{Y_i}_{\text{Prediction}} = \overbrace{\boxed{b}}^{\text{Bias}} + \underbrace{\boxed{w}}_{\text{Weight}} \overbrace{X_i}^{\text{Feature}}$$

$$\{X, Y\}_{n=1}^{N}$$

1-dim Feature

$Y = b + wX$

Geometric shape: Line

F-dim Feature

$Y = w_0 + w_1X_1 + w_2X_2 + \ldots w_fx_f$

What is the geometric shape of f = 2 ?

(1) Least Squares: F-dim Features

- **Input:** $x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$

  "features"
  "covariates"
  "predictors"
  "attributes"

  Entries can be real-valued, or other numeric types (e.g. integer, binary)

$$\tilde{X} = \begin{bmatrix} x_{11} & \ldots & x_{1F} & 1 \\ x_{21} & \ldots & x_{2F} & 1 \\ & & \ldots & \\ x_{N1} & \ldots & x_{NF} & 1 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- **Output:** $\hat{y}(x_i) \in \mathbb{R}$

  Scalar value like 3.1 or -133.7

  "responses"
  "labels"

(1) Least Squares: F-dim Features

Parameters:

weight vector $w = [w_1, w_2, \ldots w_F]$

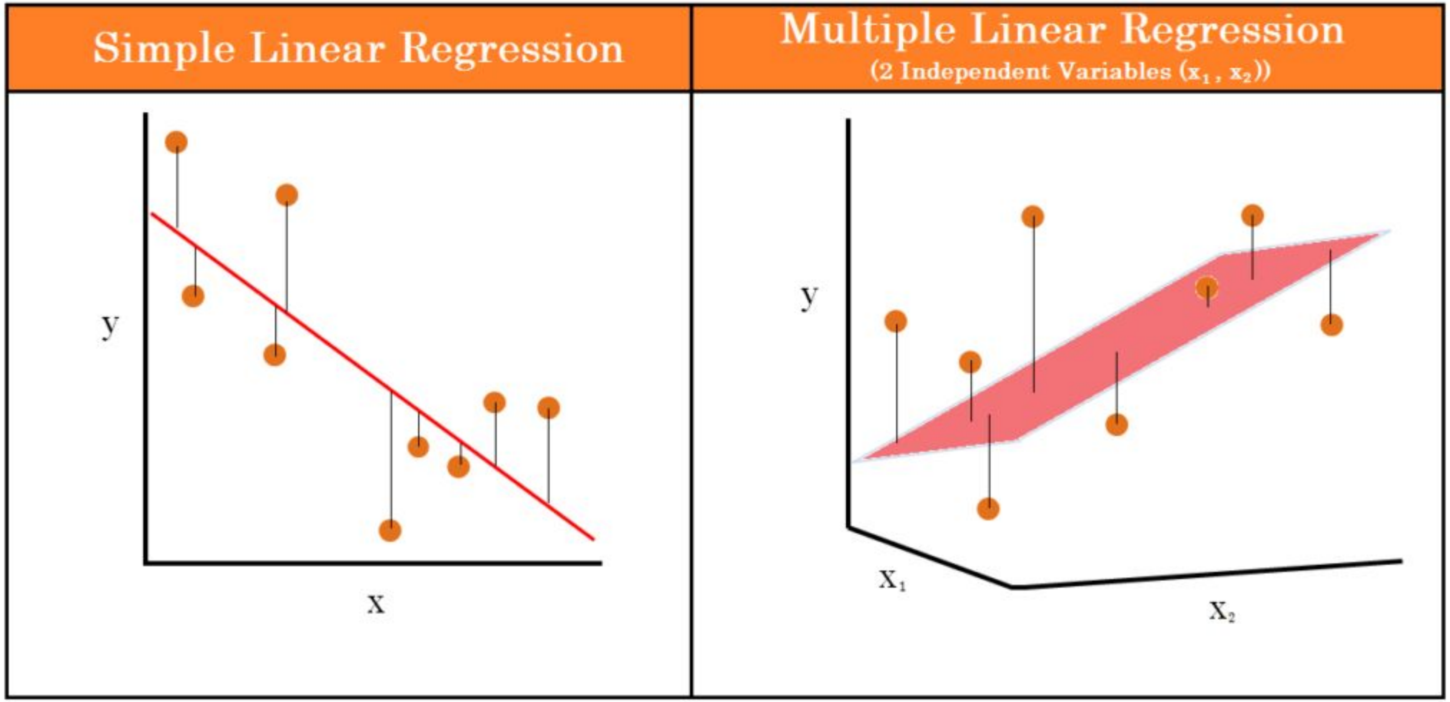
bias scalar b Or w_0

Prediction:

$$\hat{y}(x_i) \triangleq \sum_{f=1}^{F} w_f x_{if} + b$$

$$Y = w_0 + w_1 X_1 + w_2 X_2 + \ldots w_f x_f$$

(1) Least Squares: F-dim Features



https://medium.com/@thaddeussegura/multiple-linear-regression-in-200-words-data-8bdbcef34436

(1) Least Squares: F-dim Features

- **Input: Pairs of features and labels/responses**

$$\{x_n, y_n\}_{n=1}^{N}$$

- **Output:** $\hat{y}(\cdot) : \mathbb{R}^F \to \mathbb{R}$

(1) Least Squares: F-dim Features

- Solution 1:
  - Closed Form/Mathematical
- Derivation steps:

1. Compute gradient of objective wrt each entry of w, and wrt scalar b (F+1 total expressions)

2. Set all gradients equal to zero and solve for w and b (F+1 equations, F+1 unknowns)

$$\theta = [b \ w_1 \ w_2 \dots w_F]$$

$$\tilde{x}_n = [1 \ x_{n1} \ x_{n2} \dots x_{nF}]$$

$$\hat{y}(x_n, \theta) = \theta^T \tilde{x}_n$$

$$J(\theta) \triangleq \sum_{n=1}^{N} (y_n - \hat{y}(x_n, \theta))^2$$

Linear Regression: (1) Least Squares

- Task: Training
- Model Function: Multidimensional
- Cost Function: Sum of Squared Errors
- Error:
  - distance in multidimensions
- Learning Algorithm: Linear Least Square
  - Output Model:
    - Values of w and b which minimize the cost function on the training set
    - Values of θ compact form

Notebooks

https://www.youtube.com/watch?v=gj4g7CzDzJE

10 Minute break

Gradient Descent

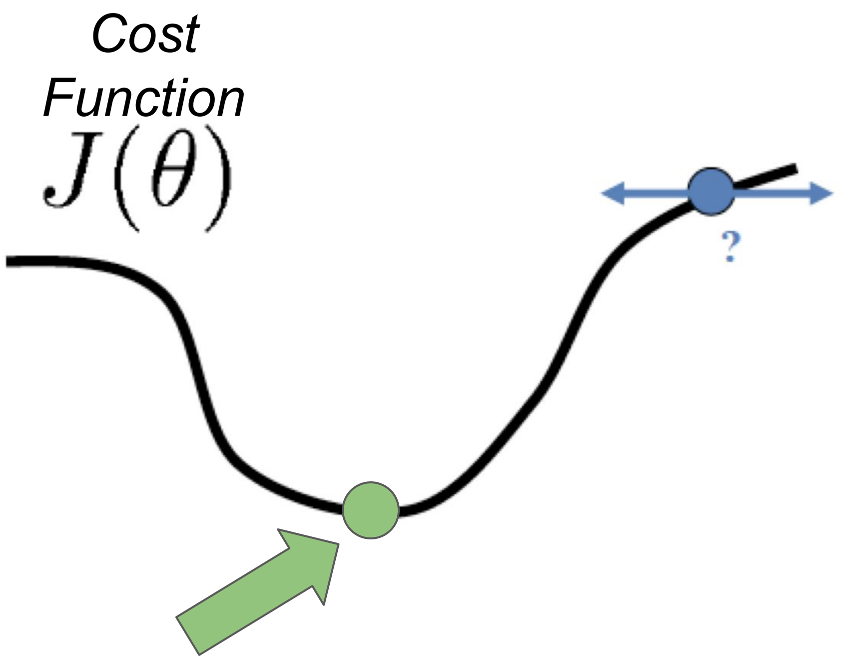
https://www.youtube.com/watch?v=gj4g7CzDzJE

Gradient Descent

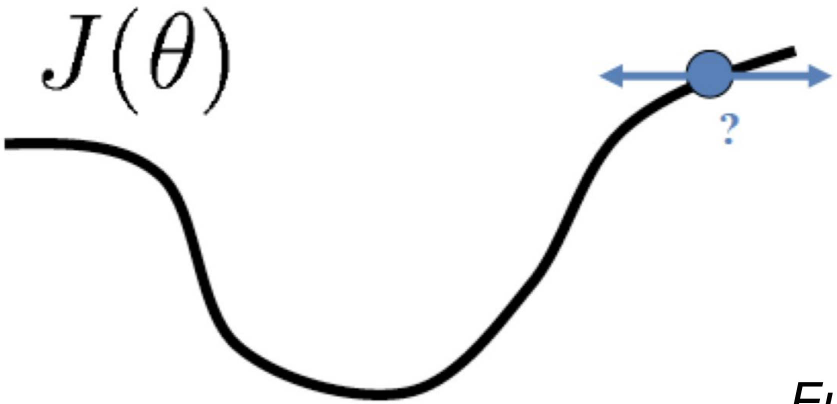
- Closed form solution is computationally expensive with large number of feature/training data
- Other methods such as Gradient Descent are more suitable in this case

Gradient Descent

- Minimization of the function J(θ) means:
    - The value of θ that makes J equals zero or close to zero
    - To visualize this concept, it is the lowest point of the cost function curve

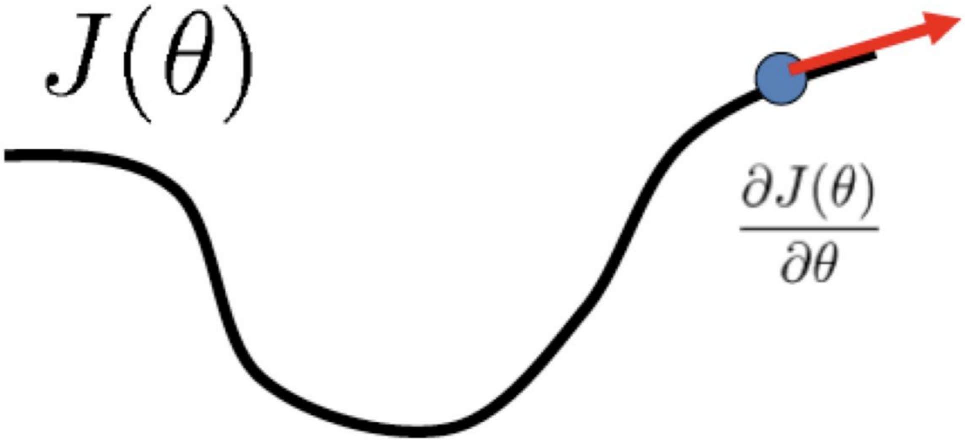


Gradient Descent

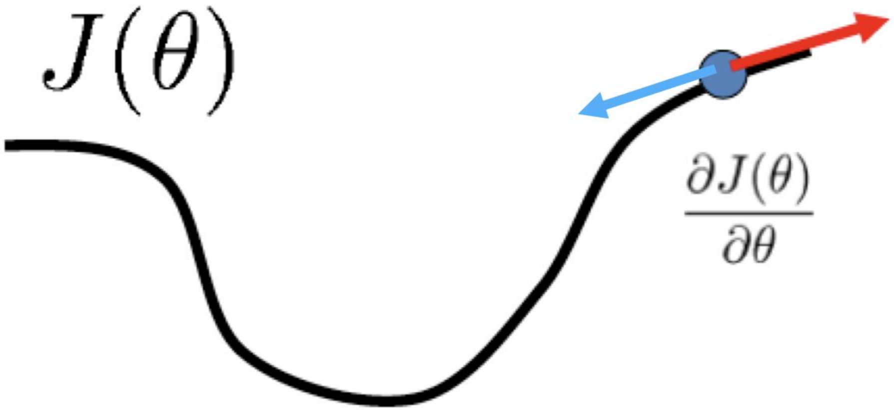


Cost Function

Gradient Descent



Gradient Descent



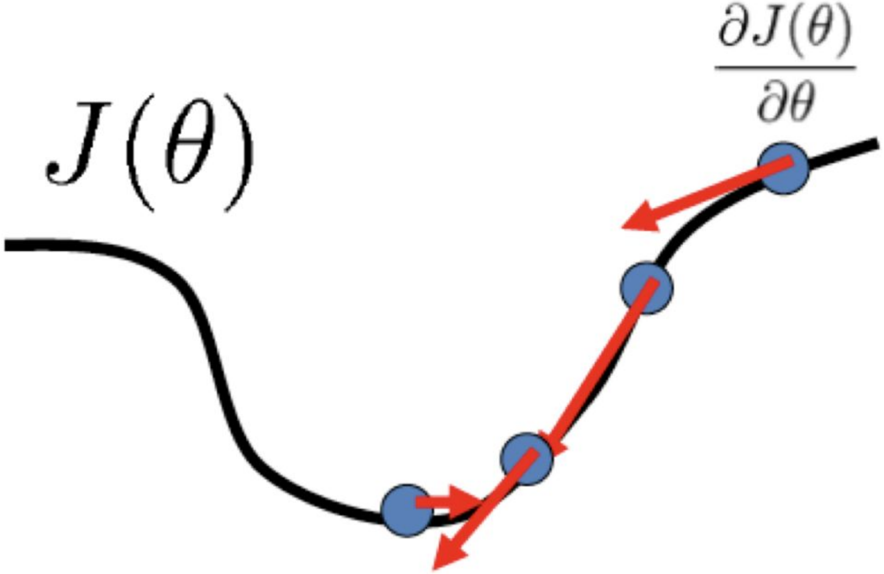
Gradient Descent

input: initial $\theta \in \mathbb{R}$

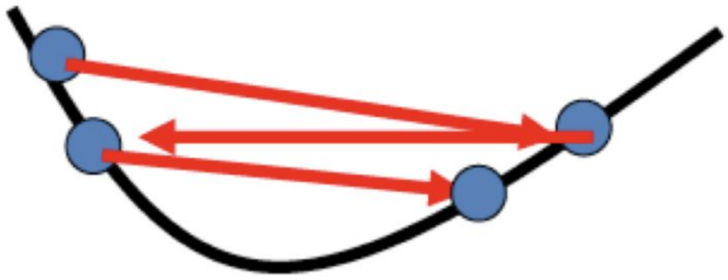
input: step size $\alpha \in \mathbb{R}_+$

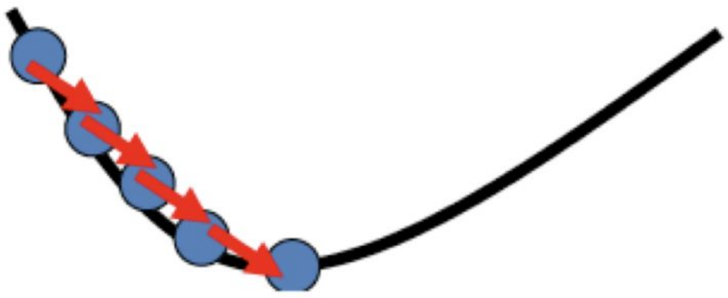
while not converged:

$$\theta \leftarrow \theta - \alpha \frac{d}{d\theta} J(\theta)$$



Gradient Descent





Gradient Descent

- Simple and usually effective: pick small constant

$$\alpha = 0.01$$

- Improve: **decay** over iterations

$$\alpha_t = \frac{C}{t} \qquad \alpha_t = (C + t)^{-0.9}$$

- Improve: Line search for best value at each step

Gradient Descent

How to assess convergence?

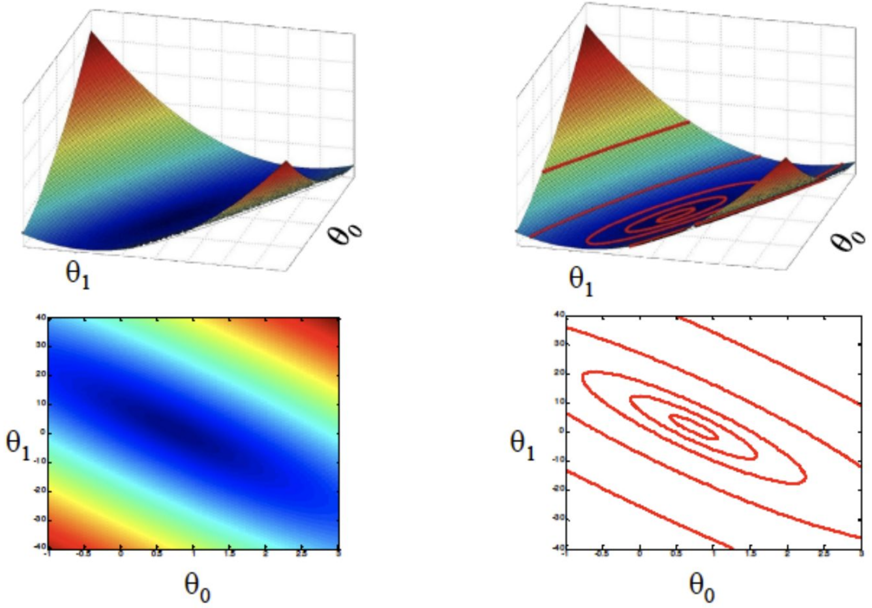
- Ideal: stop when derivative equals zero
- Practical heuristics: stop when …
  - when change in loss becomes small

$$|J(\theta_t) - J(\theta_{t-1})| < \epsilon$$

  - when step size is indistinguishable from zero

$$\alpha|\frac{d}{d\theta}J(\theta)| < \epsilon$$

Gradient Descent



“Level set” contours : all points
with same function value

Good Luck!