CS60050
Machine Learning

Linear Regression

Some slides from Matt Gormley (CMU), Dr. Yaser S. Abu-Mostafa (CalTech, USA),
Dr. Ambedkar Dukipatti (CSA, IISC)

Somak Aditya
Assistant Professor

Sudeshna Sarkar

Department of CSE, IIT Kharagpur
August 9, 2023

Dataset of living area and price of houses in a city

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮
5000	?

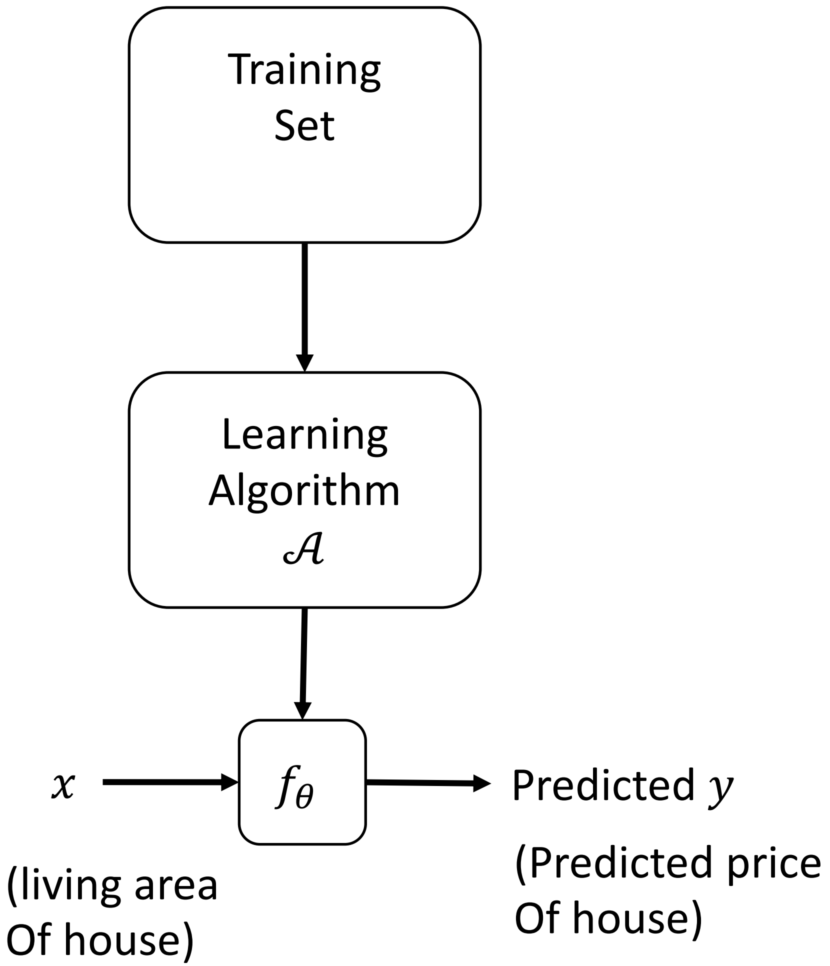
- This is a training set.
- How can we learn to predict the prices of houses of other sizes in the city, as a function of their living area?
- Example of supervised learning
  - If $y \in \mathbb{R}$, then its "regression"

Dataset of living area and price of houses in a city

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$

- m = number of **training examples**
- x_i = input variables / features
- y_i = output variables / "target" variables
- (x_i, y_i) - ith training example of the training set

How to use the training set?



- Learn a function $f(x)$, so that $f(x)$ is a good predictor for the corresponding value of y
- f: hypothesis function

How to represent hypothesis? (linear?)

$$\hat{y} = f_\theta(x) = \theta_0 + \theta_1 x$$

- θ_i are **parameters**
- $\boldsymbol{\theta}$: vector of all the parameters

- We assume
  - y is a linear function of x
- How to learn the values of the parameters θ_i ?

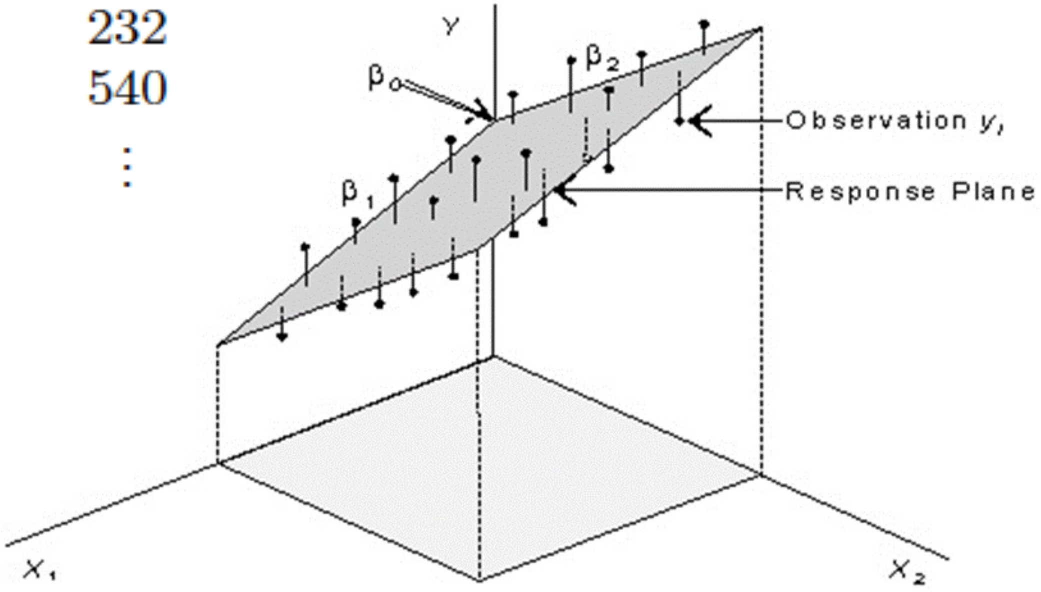
Digression: What about Multi-variate case?

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

$$f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Multivariate Regression

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



- n features
- m training examples
- $(x^{(i)}, y^{(i)})$: ith training example

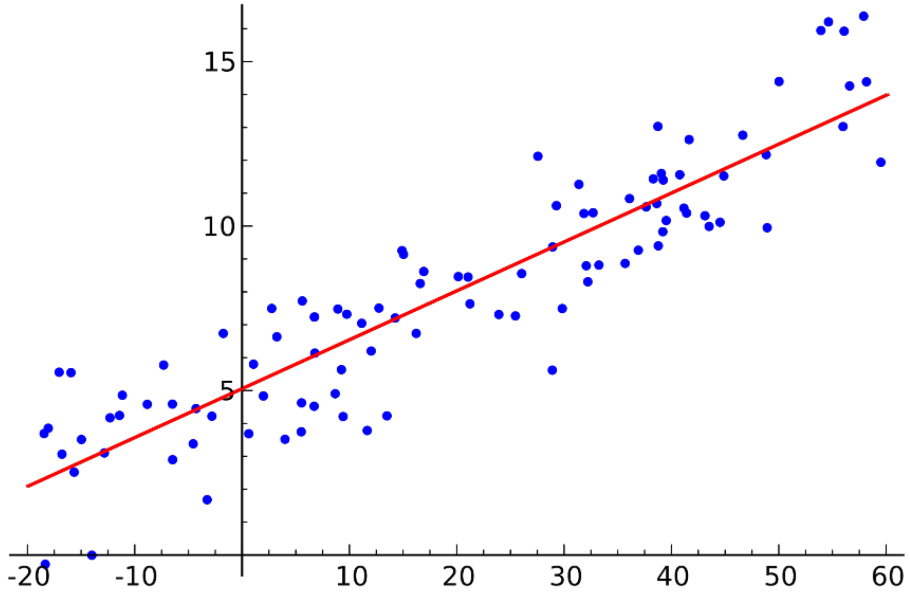
- $y = f_\theta(\mathrm{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Intuition of hypothesis function

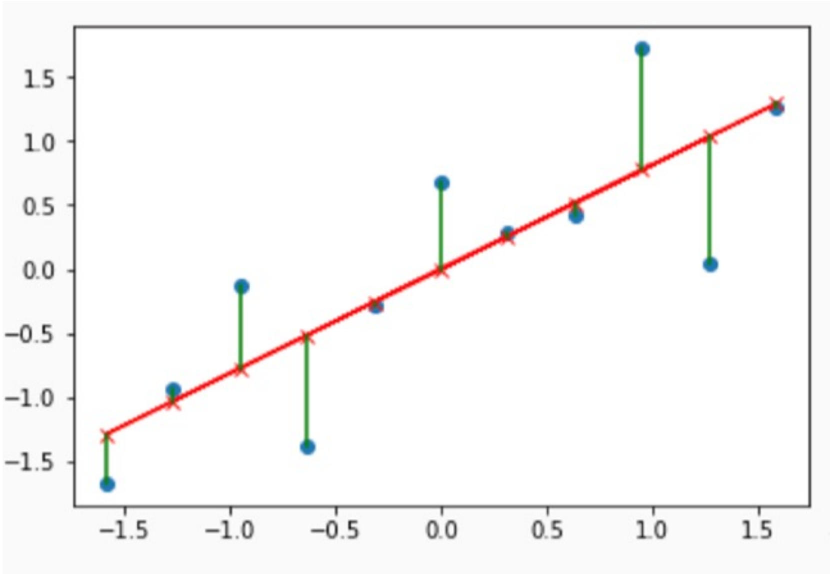
$$f_\theta(\mathrm{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Two equivalent questions:

1. Which is the best straight line to fit the data?
2. How to learn the values of the parameters θ_i ?



Cost function



$$e^{(i)} = \widehat{y^{(i)}} - y^{(i)} = f_\theta\left(x^{(i)}\right) - y^{(i)}$$

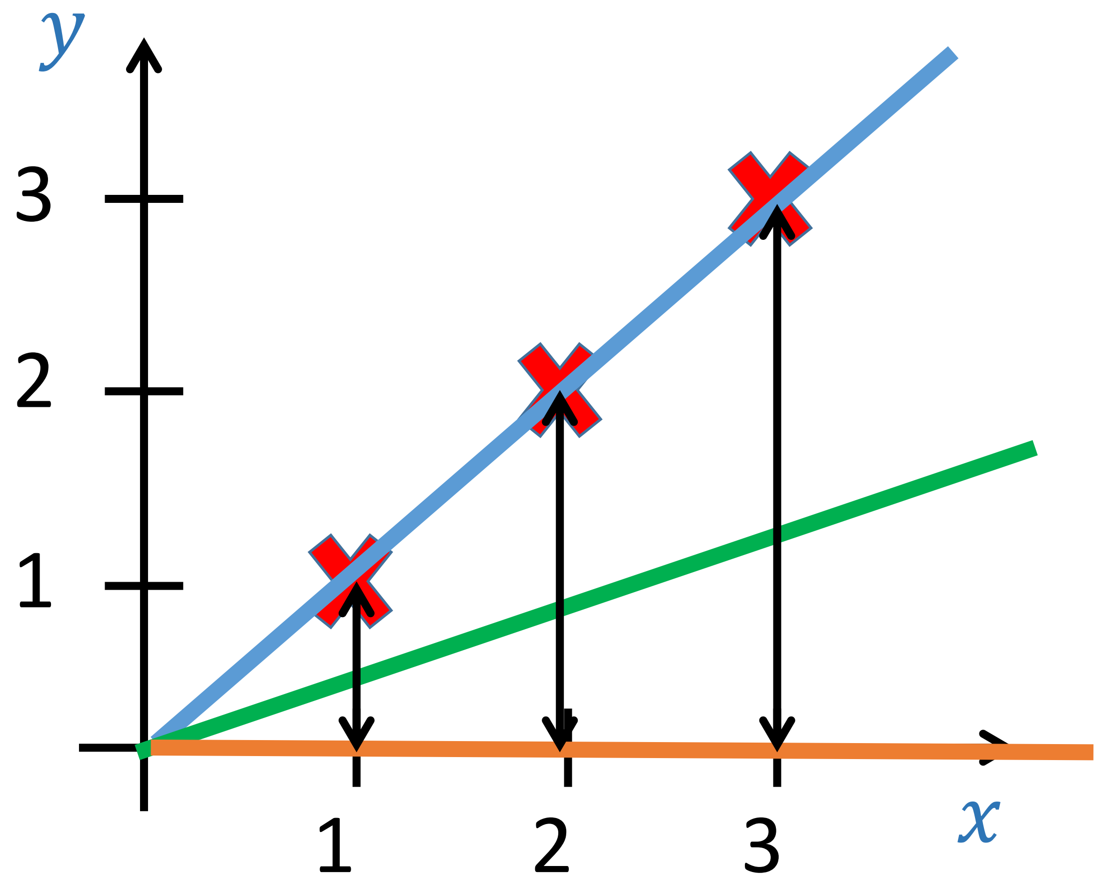
prediction error for ith training example

$$loss(\bar{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

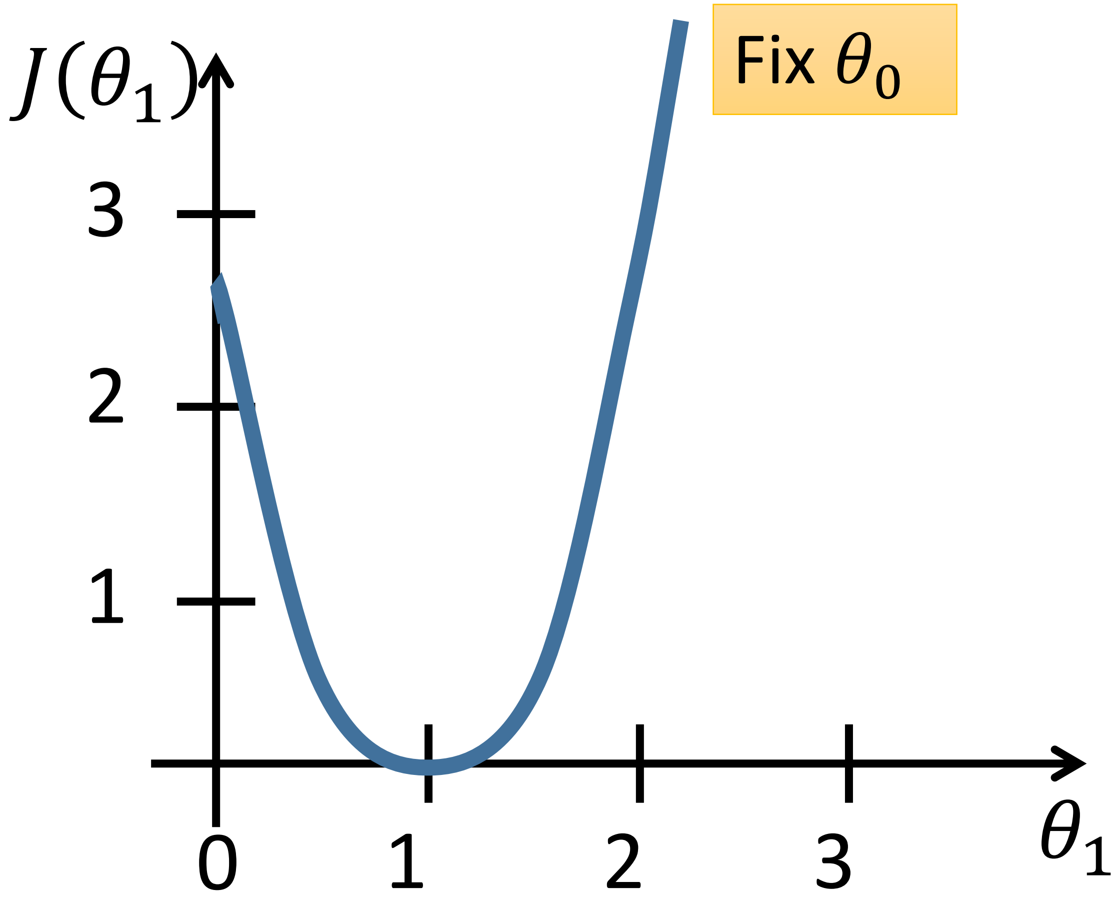
Choose parameters $\bar{\theta}$ so that

$loss(\bar{\theta})$ is minimized

$f_\theta(x)$: function of x for fixed θ

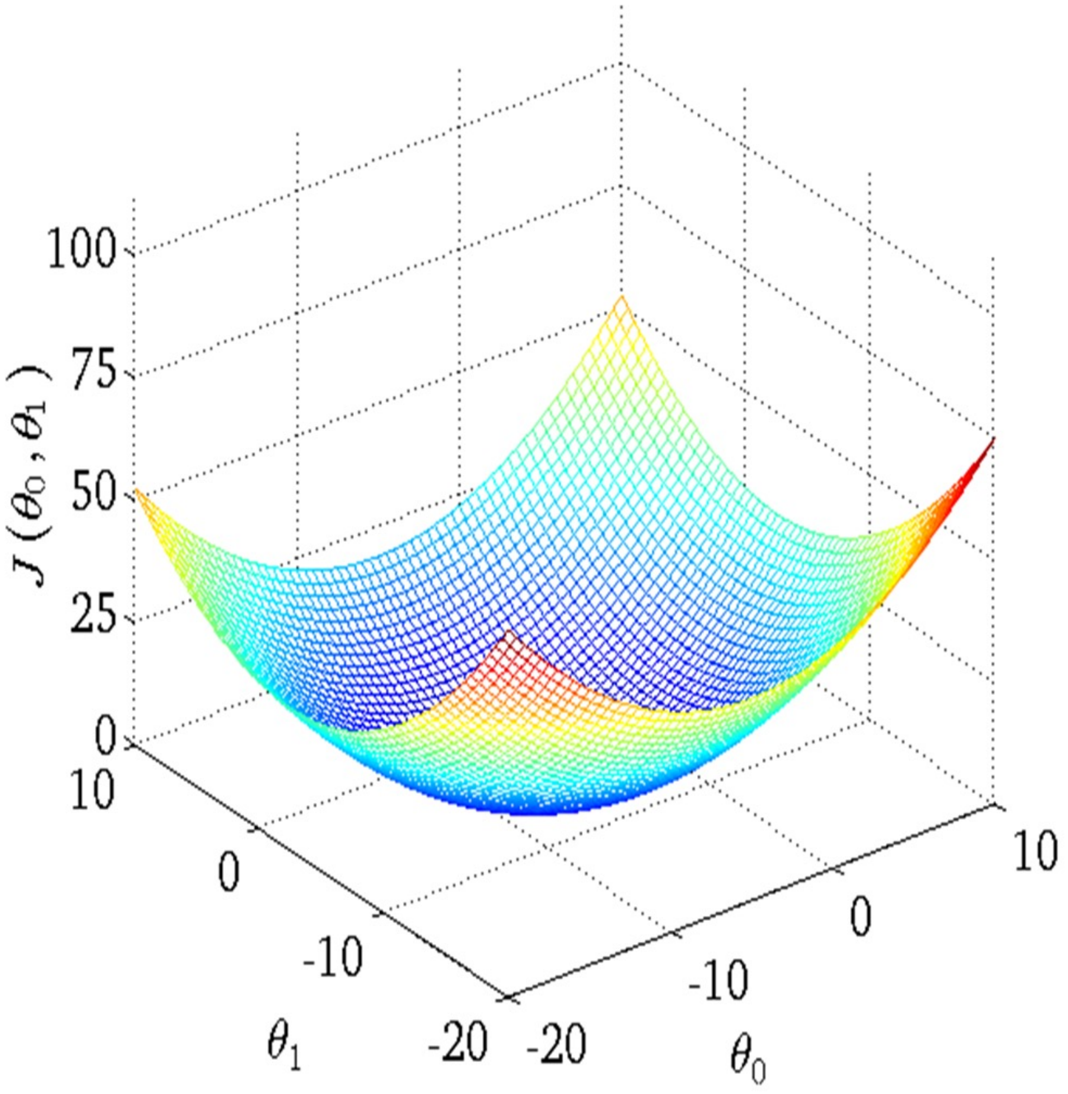


$loss(\theta)$, function of θ_0, θ_1



Cost Function

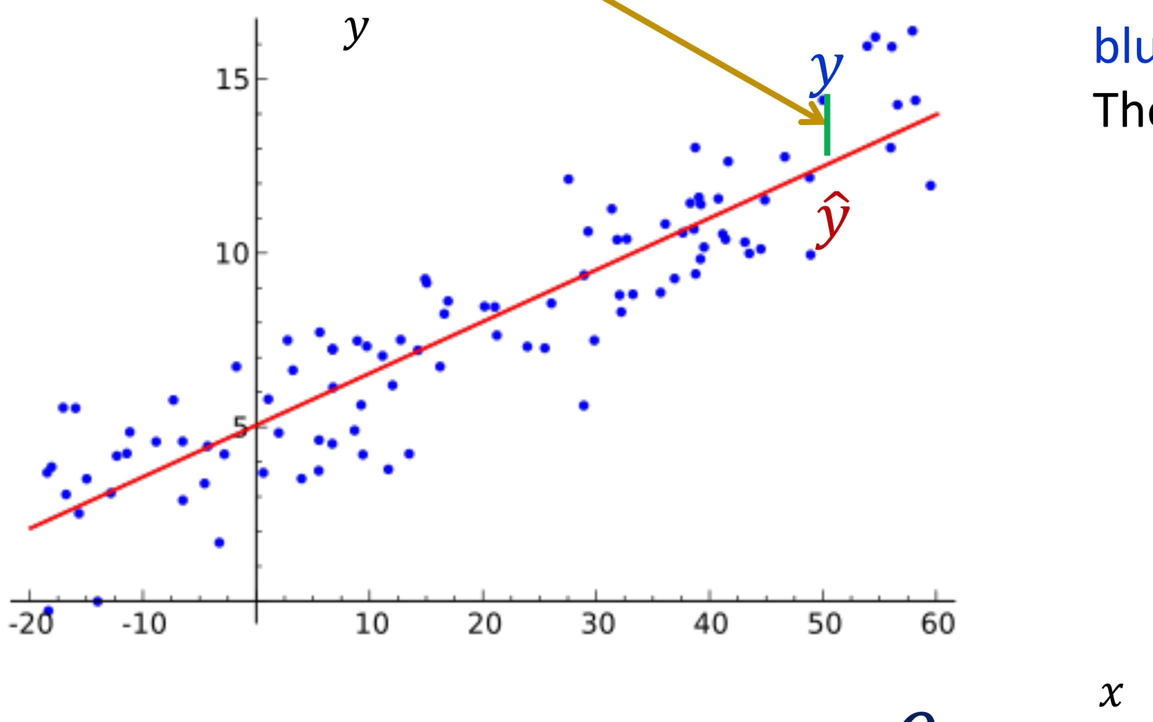
When loss is a function of both θ_0 and θ_1



Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x$$

The loss is the squared loss $L_2(\hat{y}, y) = (\hat{y} - y)^2$



Data (x, y) pairs are the blue points.

The model is the red line.

Optimization objective: Find model parameters θ that will minimize the loss.

Linear Regression

The total loss across all points is

$$L = \sum_{i=1}^{m} \left(\widehat{y^{(i)}} - y^{(i)}\right)^2$$

$$= \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{N} \sum_{i=1m} \left(f\left(x^{(i)};\ \theta\right) - y^{(i)}\right)^2$$

We want the optimum values of θ_0, θ_1 that will minimize the sum of squared errors. Two approaches:

1. Analytical solution via mean squared error
2. Iterative solution via MLE and gradient ascent

Linear Regression: Analytic Solution

Since the loss is differentiable, we set

$$\frac{dL}{d\theta_0} = 0 \quad \text{and} \quad \frac{dL}{d\theta_1} = 0$$

We want the loss-minimizing values of $\boldsymbol{\theta}$, so we set

$$\frac{dL}{d\theta_1} = 0 = 2\theta_1 \sum_{i=1}^{N} \left(x^{(i)}\right)^2 + 2\theta_0 \sum_{i=1}^{N} x^{(i)} - 2\sum_{i=1}^{N} x^{(i)} y^{(i)}$$

$$\frac{dL}{d\theta_0} = 0 = 2\theta_1 \sum_{i=1}^{N} x^{(i)} + 2\theta_0 N - 2\sum_{i=1}^{N} y^{(i)}$$

These being linear equations of θ, have a unique closed form solution

$$\theta_1 = \frac{m \sum_{i=1}^{m} y^{(i)} x^{(i)} - \left(\sum_{i=1}^{m} x^{(i)}\right)\left(\sum_{i=1}^{m} y^{(i)}\right)}{m \sum_{i=1}^{m} (x^{(i)})^2 - \left(\sum_{i=1}^{m} x^{(i)}\right)^2}$$

$$\theta_0 = \frac{1}{m}\left(\sum_{i=1}^{m} y^{(i)} - \theta_1 \sum_{i=1}^{m} x^{(i)}\right)$$

Learning as Optimization Problem

Hypothesis: $f_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $\ell oss(\theta) = \sum_{n=1}^{N} \left(y_n - f_\theta(x_n)\right)^2$

Goal: $\min_{\theta_0,\theta_1} loss(\theta_0, \theta_1)$

Linear Regression: Analytic Solution

$$L = \sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

Since the loss is differentiable, we set $\frac{dL}{d\theta_0} = 0$ and $\frac{dL}{d\theta_1} = 0$

$$\frac{dL}{d\theta_1} = 0 = 2\theta_1 \sum_{i=1}^{N}\left(x^{(i)}\right)^2 + 2\theta_0 \sum_{i=1}^{N} x^{(i)} - 2\sum_{i=1}^{N} x^{(i)} y^{(i)}$$

$$\frac{dL}{d\theta_0} = 0 = 2\theta_1 \sum_{i=1}^{N} x^{(i)} + 2\theta_0 N - 2\sum_{i=1}^{N} y^{(i)}$$

There is a unique closed form solution

$$\theta_1 = \frac{m\sum_{i=1}^{m} y^{(i)} x^{(i)} - \left(\sum_{i=1}^{m} x^{(i)}\right)\left(\sum_{i=1}^{m} y^{(i)}\right)}{m\sum_{i=1}^{m}(x^{(i)})^2 - \left(\sum_{i=1}^{m} x^{(i)}\right)^2}$$

$$\theta_0 = \frac{1}{m}\left(\sum_{i=1}^{m} y^{(i)} - \theta_1 \sum_{i=1}^{m} x^{(i)}\right)$$

Multivariate Linear Regression

$x \in \mathcal{R}^d$

$$\hat{y} = f(x;\ \theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \dots + \theta_d x_d$$

Define $x_0 = 1$ $\qquad f(x;\ \theta) = \theta^T \mathbf{x}$

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} = \begin{bmatrix} {x_0}^{(1)} & {x_1}^{(1)} & {x_2}^{(1)} & \cdots & {x_d}^{(1)} \\ {x_0}^{(2)} & {x_1}^{(2)} & {x_2}^{(2)} & \cdots & {x_d}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {x_0}^{(m)} & {x_1}^{(m)} & {x_2}^{(m)} & \cdots & {x_d}^{(m)} \end{bmatrix} \qquad \hat{\mathrm{y}} = \mathbf{X\theta}$$

Cost Function:

$$loss(\boldsymbol{\theta}) = loss(\theta_0, \theta_1, \dots, \theta_d) = \frac{1}{m}\left(\boldsymbol{\theta}^T \mathrm{x}^{(i)} - y^{(i)}\right)^2$$

Multivariate Linear Regression

$$loss(\boldsymbol{\theta}) = \frac{1}{m}\sum_i \left(\boldsymbol{\theta}^T \mathrm{x}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{m}(\mathbf{X\theta} - \mathbf{y})^T(\mathbf{X\theta} - \mathbf{y})$$

$$J(\boldsymbol{\theta}) = \frac{1}{m}\left((\mathbf{X\theta})^T - \mathbf{y}^T\right)(\mathbf{X\theta} - \mathbf{y})$$

$$= \frac{1}{m}\{(\mathbf{X\theta})^T\mathbf{X\theta} - (\mathbf{X\theta})^T\mathbf{y} - \mathbf{y}^T\mathbf{X\theta} + \mathbf{y}^T\mathbf{y}\}$$

$$loss(\boldsymbol{\theta}) = \frac{1}{m}\{\theta^T(X^TX)\theta - \theta^T X^T y - y^T X\,\theta + y^T Y\}$$

$$= \frac{1}{m}\{\theta^T(X^TX)\theta - (X^Ty)^T\theta - (X^Ty)^T\,\theta + y^T Y\}$$

$$= \frac{1}{m}\{\theta^T(X^TX)\theta - 2(X^Ty)^T\theta + y^T Y\}$$

Multivariate Linear Regression

- Equating the gradient of the cost function to 0,

$$\nabla_\theta loss(\,\boldsymbol{\theta}\,) = \frac{1}{m}\{2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} + 0\} = 0$$

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}^T\mathbf{y} = 0$$

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X}\,)^{-1}\mathbf{X}^T\mathbf{y}$$

This gives a closed form solution, but another option is to use iterative solution

$$\frac{\partial loss(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(f_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j{}^{(i)}$$

Partial derivatives

- Let $y = f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$ be a multivariate function with *n* variables
  - The mapping is $f: \mathbb{R}^n \to \mathbb{R}$
- The ***partial derivative*** of *y* with respect to its *i*th parameter x_i is

$$\frac{\partial y}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i, \dots, x_n)}{h}$$

- To calculate $\frac{\partial y}{\partial x_i}$, we can treat $x_1, x_2, \dots, x_{i-1}, x_{i+1} \dots, x_n$ as constants and calculate the derivative of *y* only with respect to x_i
- For notation of partial derivatives, the following are equivalent:

$$\frac{\partial y}{\partial x_i} = \frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} f(\mathbf{x}) = f_{x_i} = f_i = D_i f = D_{x_i} f$$

Multidimensional derivative: Gradient

Gradient vector: The gradient of the multivariate function $f(\mathbf{x})$ with respect to the *n*-dimensional input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, is a vector of *n* partial derivatives

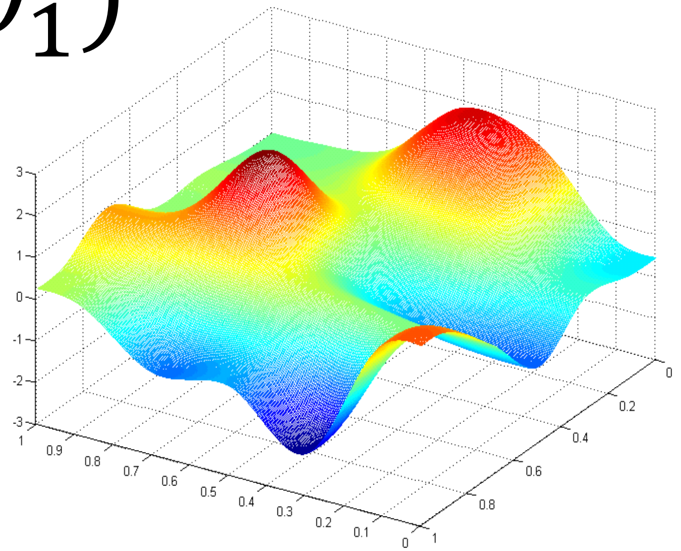
$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]^T$$

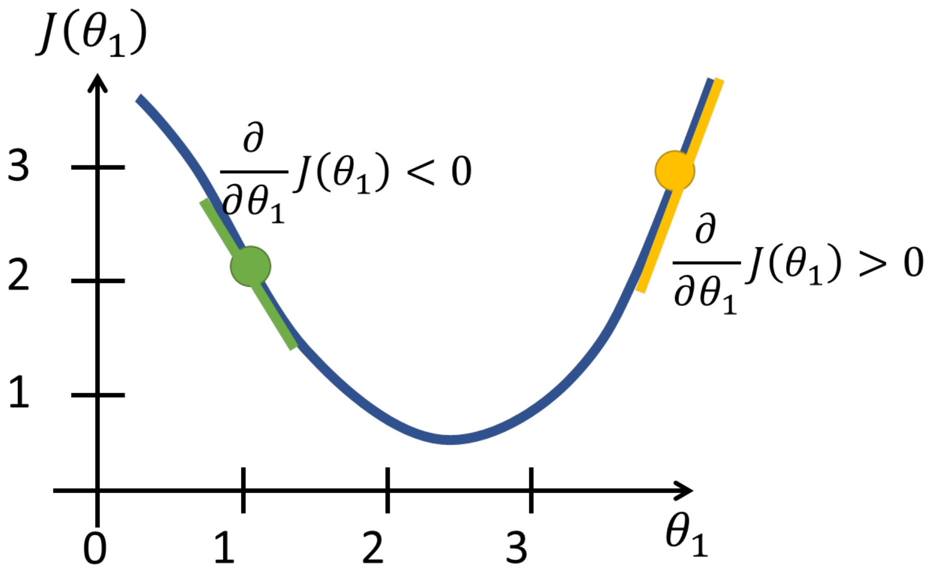
- In ML, the gradient descent algorithm relies on the opposite direction of the gradient of the loss function $\mathcal{L}$ with respect to the model parameters θ ($\nabla_\theta \mathcal{L}$) for minimizing the loss function

Minimizing cost function & Gradient Descent

Minimizing function $loss(\theta_0, \theta_1)$

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $loss(\theta_0, \theta_1)$
- until we end up at a minimum

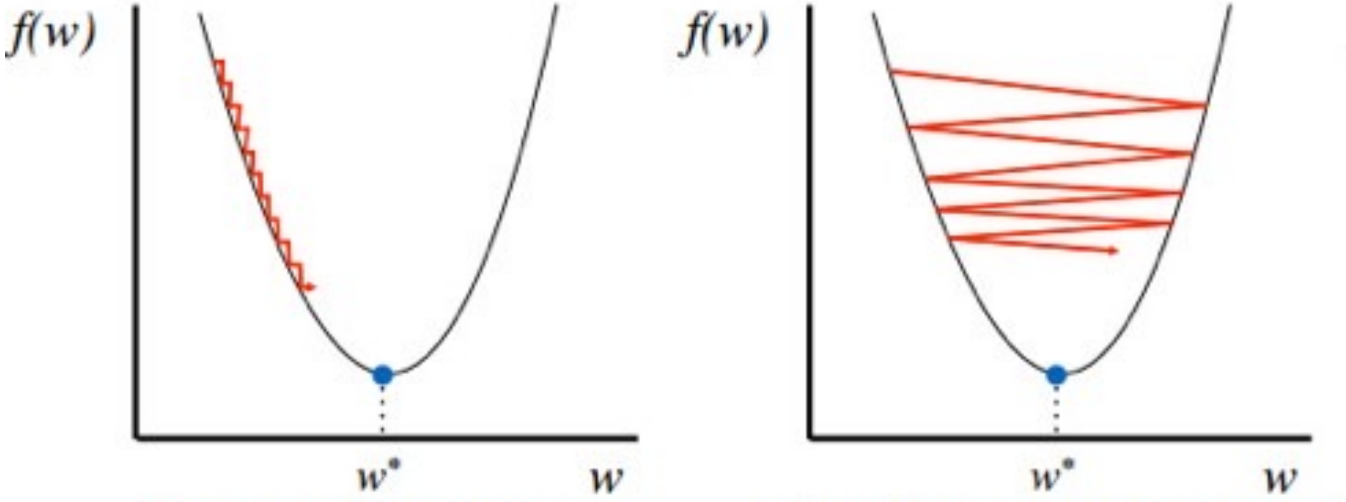




$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} loss(\theta_1)$$

Step Size α

- α Determines how quickly training loss goes down; hence “learning rate”



Too small: converge very slowly

Too big: overshoot and even diverge

Computing partial derivatives

Repeat until convergence{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} loss(\bar{\theta})$$

Equivalently

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

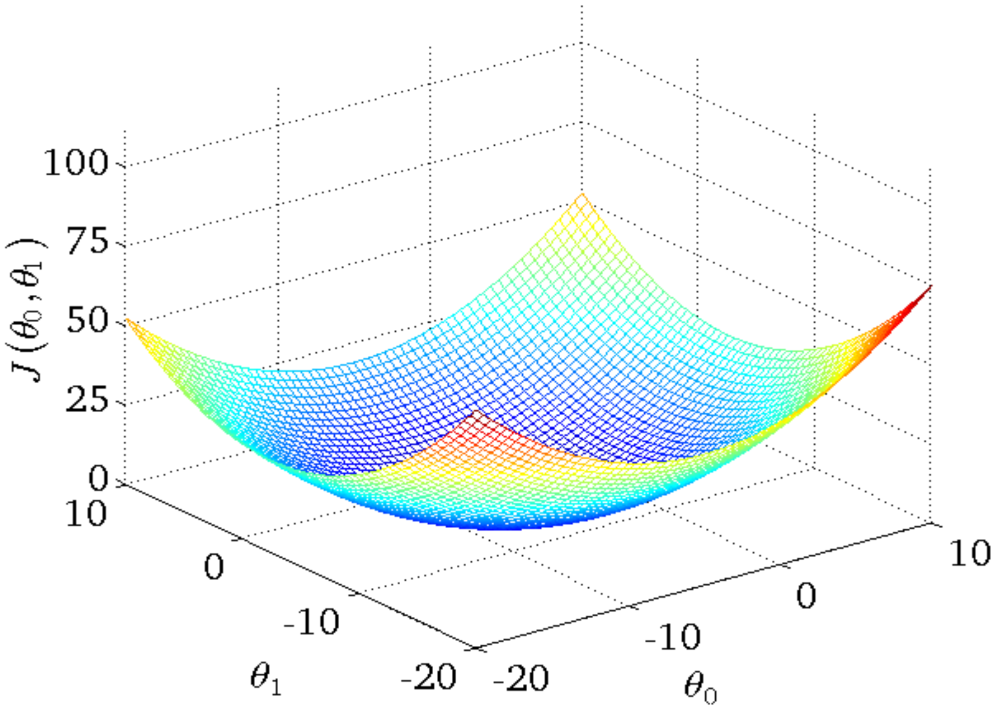
}

$$loss(\bar{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

$$\frac{\partial}{\partial \theta_j} loss(\bar{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(f_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

Convergence

- The cost function in linear regression is always a convex function – always has a single global minimum
- So, gradient descent will always converge



Batch gradient descent

"Batch": Each step of gradient descent uses all the training examples

Repeat until convergence{

m: Number of training examples

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{\boxed{m}} \left(f_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{\boxed{m}} \left(f_\theta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

}