Université Constantine 2
جامعة قسنطينة 2

Module : Machine Learning (ML – SDSI)

– Course 1 –

Chapter 1 : Supervised Learning for Regression

Habib-Ellah GUERGOUR
Faculty of NTIC / TLSI Department
Contact: habib.guergour@univ-constantine2.dz

Université Constantine 2
جامعة قسنطينة 2

Module : Machine Learning (ML – SDSI)

– Course 1 –

Chapter 1 : Supervised Learning for Regression

Habib-Ellah GUERGOUR
Faculty of NTIC / TLSI Department
Contact: habib.guergour@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
NTIC	TLSI	M1	SDSI

Goals of the Chapter

- Understand the fundamentals of **supervised learning for regression**.
- Learn key regression models: **univariate**, **multivariate**, **polynomial**, **Ridge**, and **Lasso**.
- Explore **mathematical concepts**: cost function, gradient descent, and regularization.
- Evaluate models using **performance metrics** (MSE, R^2, etc.).

Main Titles

- Introduction
- **Notation** and Key Concepts
- **Univariate** Linear Regression
- **Multivariate** Linear Regression: Scaling Up
- **Polynomial** Regression: Modeling Non-Linearity
- **Regularization** of Ridge and Lasso
- Evaluating Model Performance: **Metrics**

Introduction

Introduction to Supervised Learning for Regression

- **Supervised learning** is a type of machine learning where the model learns from labeled data, meaning each input has a corresponding output.
- The **goal** of this chapter is **to study regression**, a supervised learning task where the objective is to predict a continuous value based on input features.

How can we find a function $h_\theta(x)$ that best approximates the relationship between input variables x and the target y?

Introduction

Examples and Applications of Regression

- **House Price Prediction**: Estimate the price of a house based on size, location, and number of rooms.
- **Stock Market Forecasting**: Predict stock prices based on historical trends.
- **Medical Diagnosis**: Predict blood pressure based on patient data.
- **Sales Forecasting**: Estimate future sales based on past performance.
- …

Key Concepts and Notations

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

In supervised learning, there are four fundamental keys that form the foundation of how machine learning models learn from data:

- Dataset
- Model/Hypothesis
- Loss Function
- Optimization Algorithm

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

Example: How to get the best way? (click on the link to watch the video)

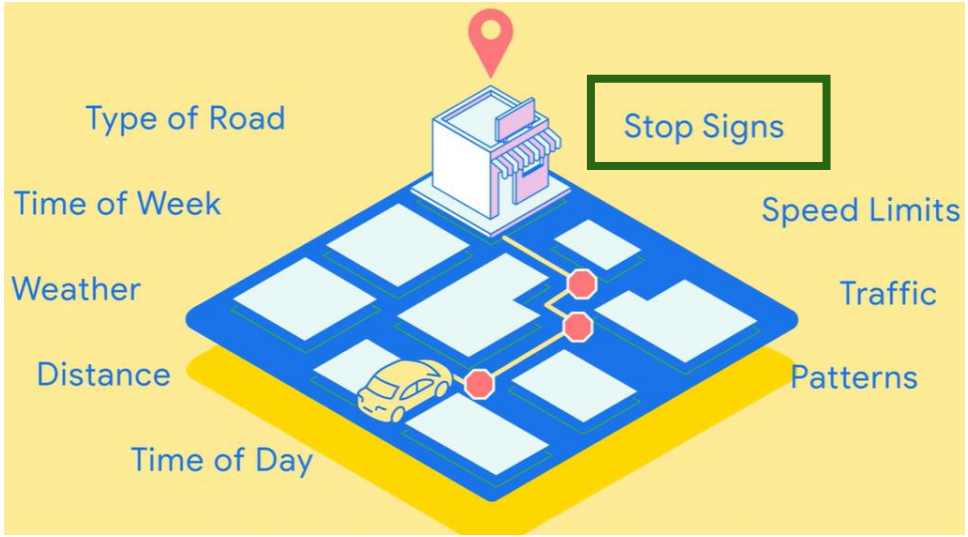
Problem

Hypothesis : Problem can be resolved by linear regression model

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

Example: How to get the best way?



Problem

Hypothesis :
Several features : Multivariate Regression Model
1 Feature : Univariate (Simple) Regression Model

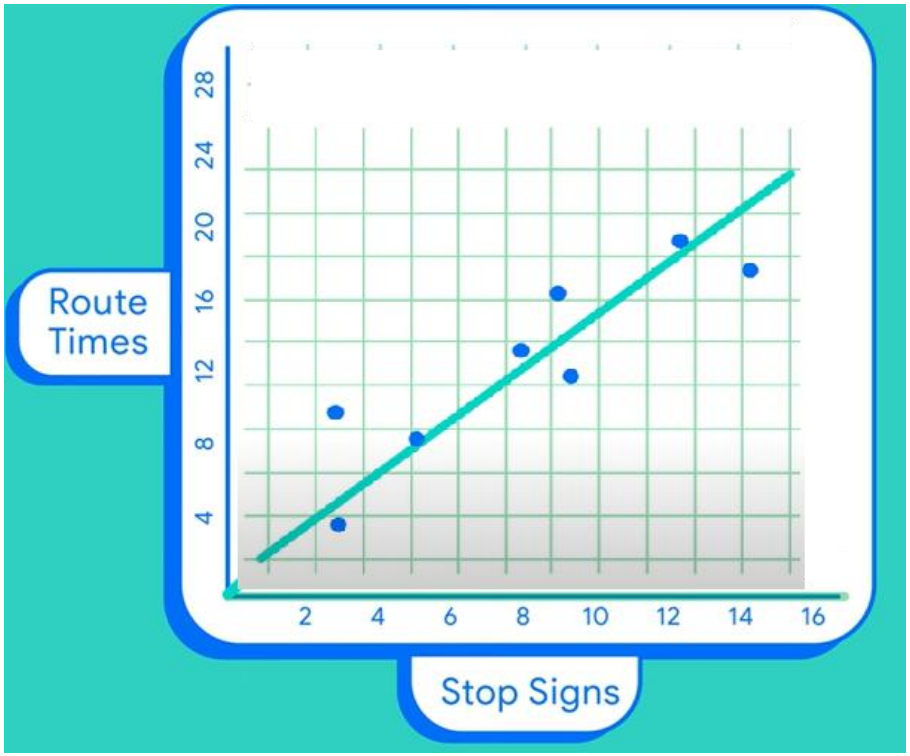
Key Concepts and Notation

Key Elements of Supervised Learning Models ?

Example: Univariate Linear Regression

Stop Signs	Route Times (Minutes)
1	4
4	7
8	12
11	15
15	17

Dataset

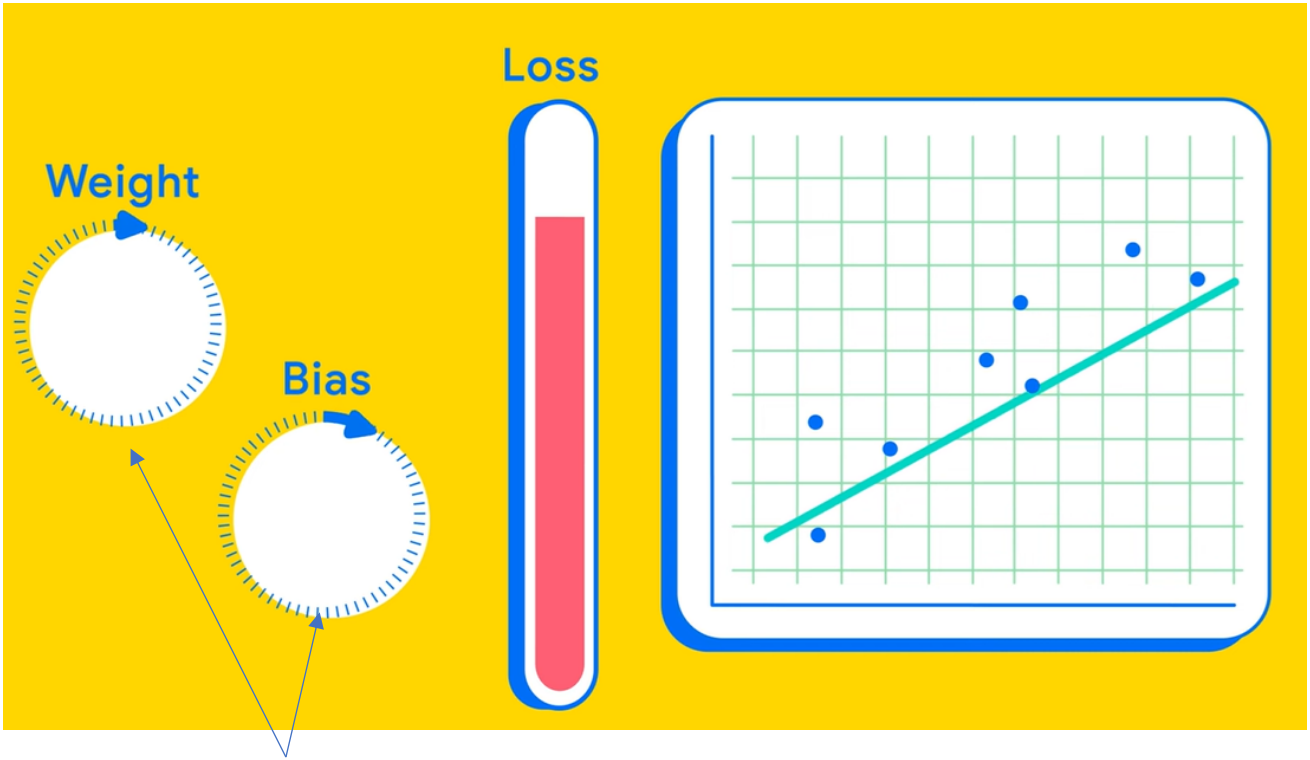


Linear Regression Model

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

Example: Univariate Linear Regression



Key Concepts and Notation

Key Elements of Supervised Learning Models ?

Example: Univariate Linear Regression

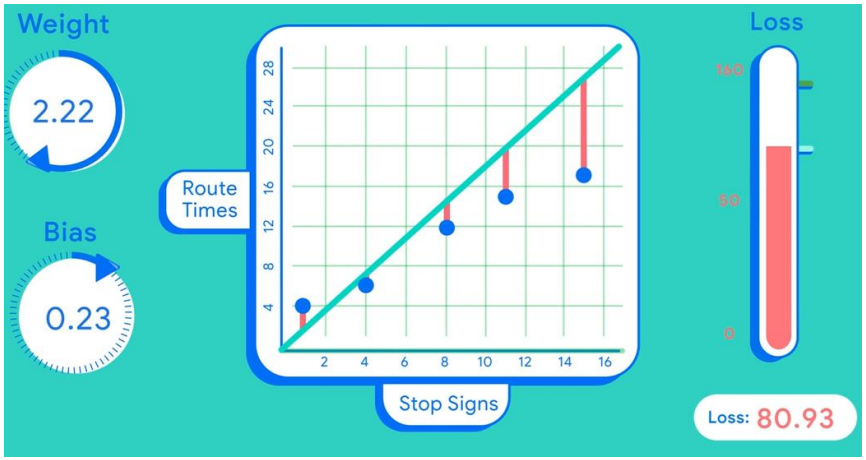
y : predicted value

Error of prediction

y : correct value

Key Concepts and Notation

Key Elements of Supervised Learning Models ?



(A)

(B)

(C)

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

1- Dataset

The collection of labeled training data, it provides the learning material for the model.

The dataset contains:

- Input features (X)
- Corresponding output labels (y)

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

1- Dataset (Notation) :

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

Where :

- $\mathcal{D}$: The dataset.
- m: The total number of examples (samples, instances).
- $x^{(i)}$: The **feature vector** of the i-th example.
- $y^{(i)}$: The **label (target value)** of the i-th example (**only in supervised learning**).

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

1- Dataset (Notation) :

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

Each sample (Example) in the dataset has n features (input variables), forming a feature vector:

$$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})^T$$

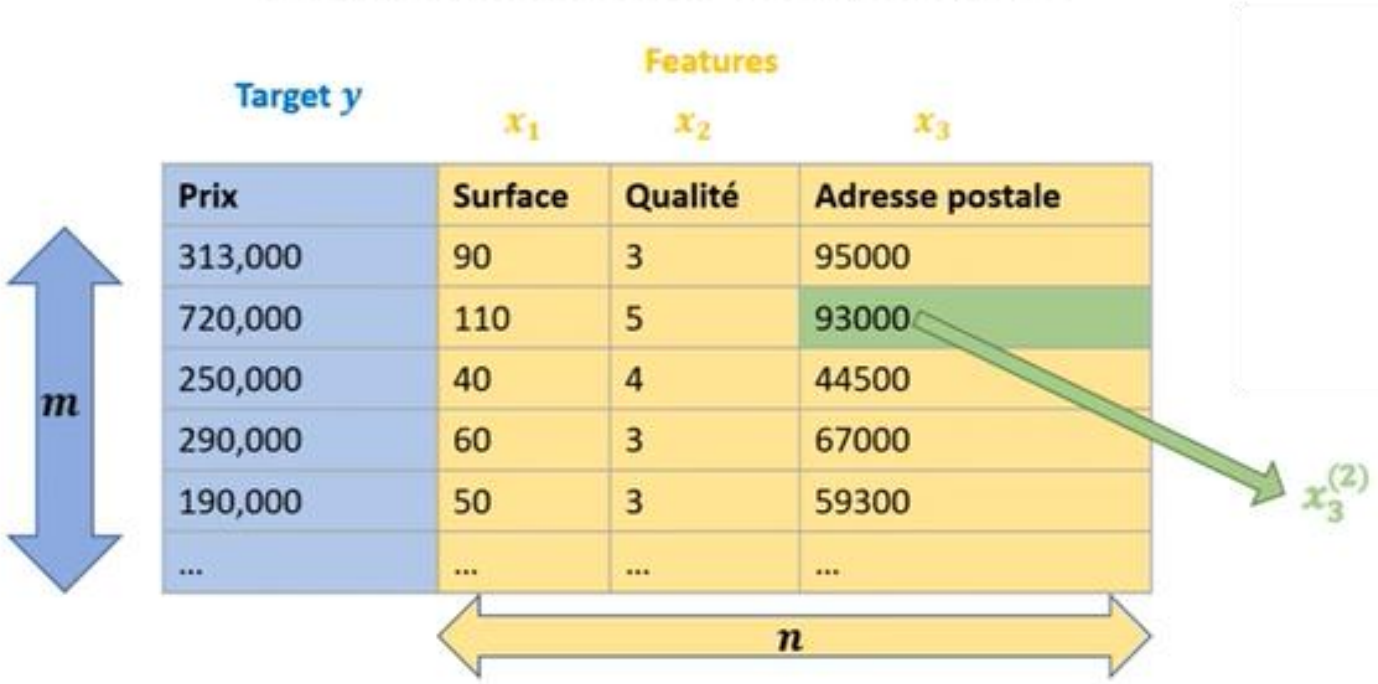
- $x^{(i)}$: The feature vector of the i-th sample.
- $x_j^{(i)}$: The j-th feature of the i-th example.
- n : The number of features (input variables).

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

1- Dataset (Example) :

Exemple de Dataset sur des appartements



Key Concepts and Notation

Key Elements of Supervised Learning Models ?

1- Dataset (Example) :

$Dataset\ (x, y)$

y	x_1	x_2	x_3	...	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	...	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	...	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	...	$x_n^{(3)}$
...	...	...	...	...	...
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	...	$x_n^{(m)}$

$vecteur\ target\ y \in \mathbb{R}^{m \times 1}$

$matrice\ features\ X \in \mathbb{R}^{m \times n}$

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{pmatrix}$$

$$X = \begin{pmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}$$

Key Concepts and Notation

Key Elements of Supervised Learning Models ?

2 - Hypothesis Function: A hypothesis function represents the model's prediction for a given input. It defines the relationship between input variables x and the predicted output y^.

$$h_\theta(x) = f(x, \theta)$$

Where:

- f is a function that defines how inputs are transformed into outputs.
- x is the feature vector (input variables).
- θ is the parameter vector (weights to be optimized).

Example of General Hypothesis Functions for linear regression :

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Key Elements of Supervised Learning Models ?

3 - Cost (Loss) Function:

A cost function quantifies how far the model's predictions are from the true values.

Regression Example (Mean Squared Error - MSE):

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Loss}(h_\theta(x^{(i)}), y^{(i)})$$

where:

- $h_\theta(x)$ is the model's hypothesis (prediction).
- θ are the parameters to optimize.
- $J(\theta)$ is the loss (error) function.

Key Elements of Supervised Learning Models ?

4 - Optimization algorithms :

Optimization algorithms help find the best parameters $\theta\theta$ that minimize the loss function $J(\theta)$, improving the model's predictions.

We need to minimize the cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Loss}(h_\theta(x^{(i)}), y^{(i)})$$

Types of Optimization Algorithms:

- Least Squares (Analytical Solution)
- Gradient Descent (GD)
- …

Univariate Linear Regression

Introduction

- **Univariate Linear Regression :**

A supervised learning algorithm that models the **relationship** between a **single input** feature x and an **output** variable y using a **straight-line equation**.

Univariate Linear Regression

Introduction

- **Goal**: Find the best-fitting straight line that predicts y given x.
  - Example: Predicting house prices based on square footage.

Target y	Features x_1	x_2	x_3
Prix	**Surface**	**Qualité**	**Adresse postale**
313,000	90	3	95000
720,000	110	5	93000
250,000	40	4	44500
290,000	60	3	67000
190,000	50	3	59300
...	...	...	...

Introduction

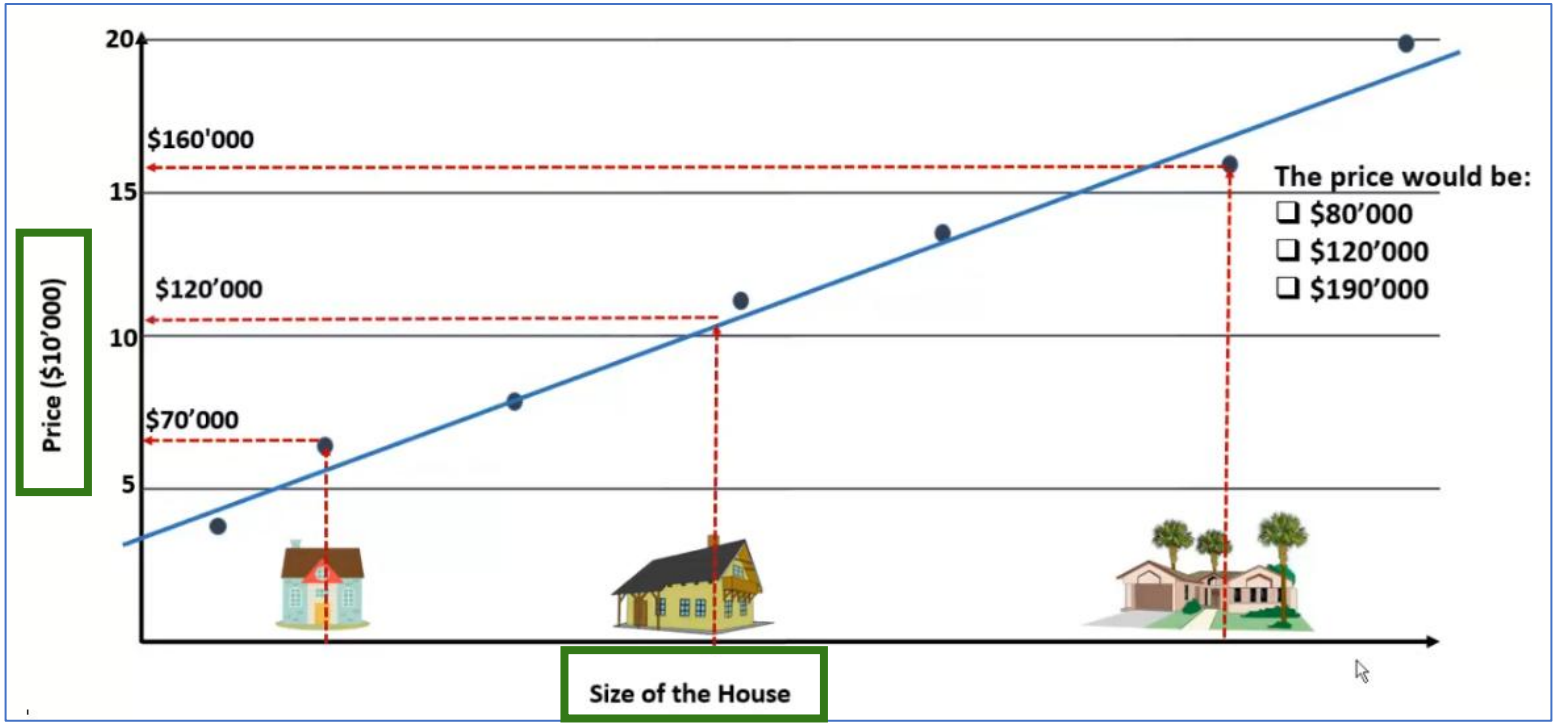
- **Goal**: Find the best-fitting straight line that predicts y given x.
  - Example: Predicting house prices based on square footage.

Target y | Features x_1 | x_2 | x_3

Prix	Surface	Qualité	Adresse postale
313,000	90	3	95000
720,000	110	5	93000
250,000	40	4	44500
290,000	60	3	67000
190,000	50	3	59300
...	...	...	...

Introduction

- **Goal**: Find the best-fitting straight line that predicts y given x.
  - Example: Predicting house prices based on square footage.



Hypothesis Function (Model)

The hypothesis function models the relationship between x and y:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

where:

- θ_0 (intercept) is the value of y when $x = 0$.
- θ_1 (slope) determines how much y changes when x increases.
- x is the **input feature** (independent variable).

Cost Function $J(\Theta)$

Why a Cost Function? We need a way to measure error between predictions and actual values.

We use Mean Squared Error (MSE) as the cost function (Erreur quadratique Moyenne)

$$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$$

Where:

$h_\Theta(x^{(i)})$ is the predicted value.

$y^{(i)}$ is the actual value.

$\frac{1}{2m}$ is used for easier differentiation.

Optimization Algorithms

Types of Optimization Algorithms:

- Least Squares (Analytical Solution)
- Gradient Descent (Approximate Solution)
- …

What is the Least Squares Method?

What is the Least Squares Method?

- An analytical approach to minimize the sum of the squared differences between observed and predicted values.

Objective:

- Find the parameters (e.g., slope and intercept) of a model that best fits the data.

What is the Least Squares Method?

Problem Definition

- Given a dataset with n samples:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

- We want to find a linear function:

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

that minimizes the difference between predicted (y^) and actual values (y).

What is the Least Squares Method?

Cost Function (Sum of Squared Errors)

The error for each point is:

$$e_i = y_i - \hat{y}_i = y_i - (\theta_0 + \theta_1 x_i)$$

The Least Squares Cost Function (Méthode des moindres carrés):

$$J(\theta_0, \theta_1) = \sum_{i=1}^{n} (y_i - (\theta_0 + \theta_1 x_i))^2$$

The **optimal** values of θ_0 and θ_1 are those that **minimize** $J(\theta_0,\theta_1)$.

Univariate Linear Regression

What is the Least Squares Method?

To find θ_0 and θ_1, we take the partial derivatives of $J(\theta_0,\theta_1)$ and set them to zero:

$$\frac{\partial J}{\partial \theta_0} = -2\sum_{i=1}^{m}(y_i - (\theta_0 + \theta_1 x_i)) = 0$$

$$\frac{\partial J}{\partial \theta_1} = -2\sum_{i=1}^{m} x_i(y_i - (\theta_0 + \theta_1 x_i)) = 0$$

Solving these equations gives:

$$\theta_1 = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{m}(x_i - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

What is the Least Squares Method?

Simplification

- The hypothesis function (model) is:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- For multiple training examples, we write it in matrix form, called Normal Equation (Closed-form Solution) :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$$

where:

- $\mathbf{y}$ is the vector of target values $(m \times 1)$.
- $\mathbf{X}$ is the **design matrix** $(m \times 2)$, including a column of ones for θ_0.
- $\boldsymbol{\theta}$ is the **parameter vector** (2×1).
- $\boldsymbol{\epsilon}$ is the error term.

What is the Least Squares Method?

Simplification

- The hypothesis function (model) is:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- For multiple training examples, we write it in matrix form, called Normal Equation (Closed-form Solution) :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix}$$

What is the Least Squares Method?

Simplification

- To minimize the sum of squared errors:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

- which in matrix form is:

$$J(\theta) = \frac{1}{2m} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$$

- Taking the derivative w.r.t. θ:

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

- The optimal solution:

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

What is the Least Squares Method?

Pros and Cons of Least Squares Method

Pros

- ✓ Fast and exact for small to medium-sized datasets.
- ✓ No need for iterative optimization.

Cons

- × Computationally expensive for large datasets (due to matrix inversion).
- × Not applicable to all models (e.g., non-linear models or models with regularization).

Gradient Descent (Alternative to Normal Equation)

What is Gradient Descent?

Gradient Descent updates Θ step by step to minimize the cost function.

- Start with an initial guess for Θ (random values or zeros).
- Compute the gradient (slope) of the cost function.
- Adjust Θ in the opposite direction of the gradient.
- Repeat until convergence (i.e., when changes in Θ become very small).

Gradient Descent (Alternative to Normal Equation)

What is Gradient Descent?

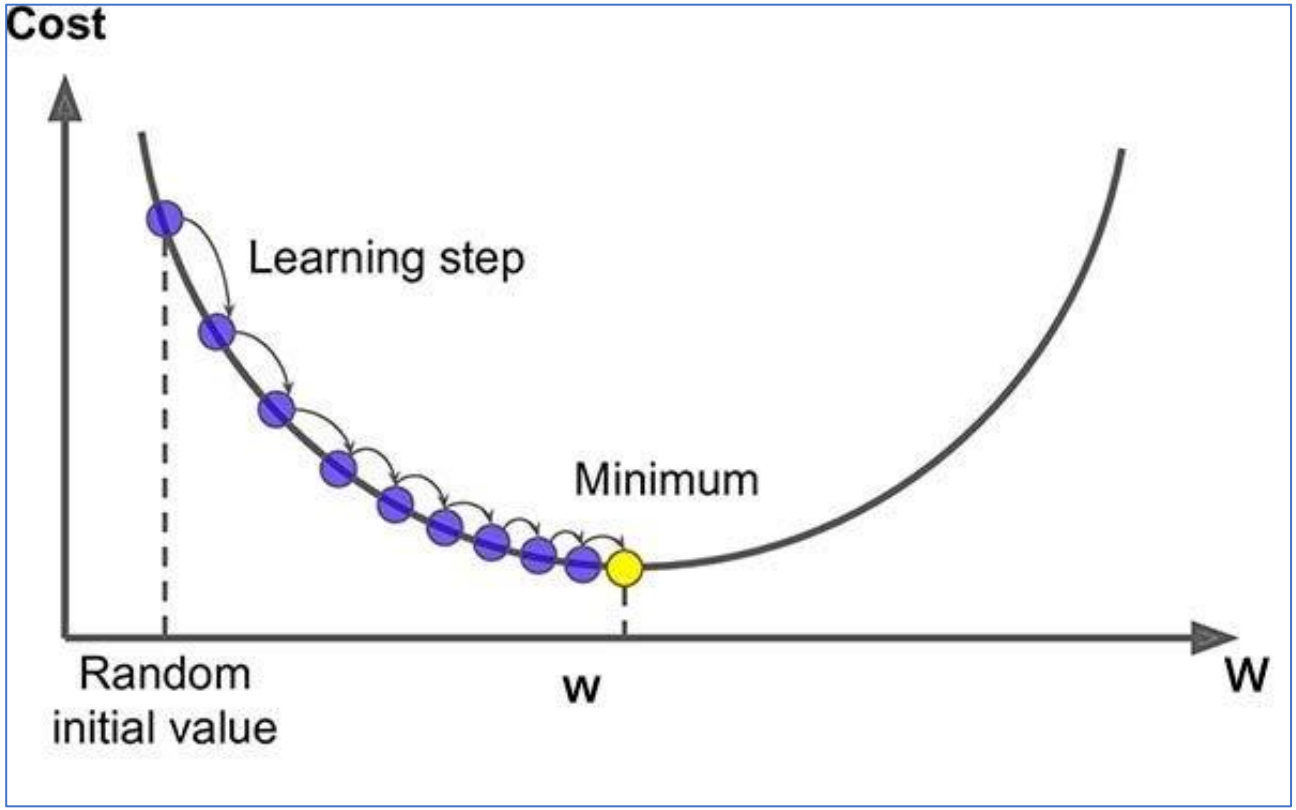
- Gradient Descent is an iterative optimization method that **updates θ step by step** in the direction of the negative gradient of the cost function:

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

- where α is the **learning rate**, which controls the step size and the second parameter is the gradient.

Gradient Descent (Alternative to Normal Equation)

What is Gradient Descent?



Gradient Descent (Alternative to Normal Equation)

Gradient Descent Algorithm

- Gradient descent updates the parameters θ_0 and θ_1 iteratively using the following update rules:

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j} \quad (j = 0..1)$$

where:

- α is the learning rate,
- $\frac{\partial J}{\partial \theta_j}$ is the gradient of the cost function w.r.t. θ_j.

Gradient Descent Algorithm

- Compute the gradients:

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_i$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0} \left[\frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2 \right]$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \left[\frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2 \right]$$

Gradient Descent (Alternative to Normal Equation)

Gradient Descent Algorithm

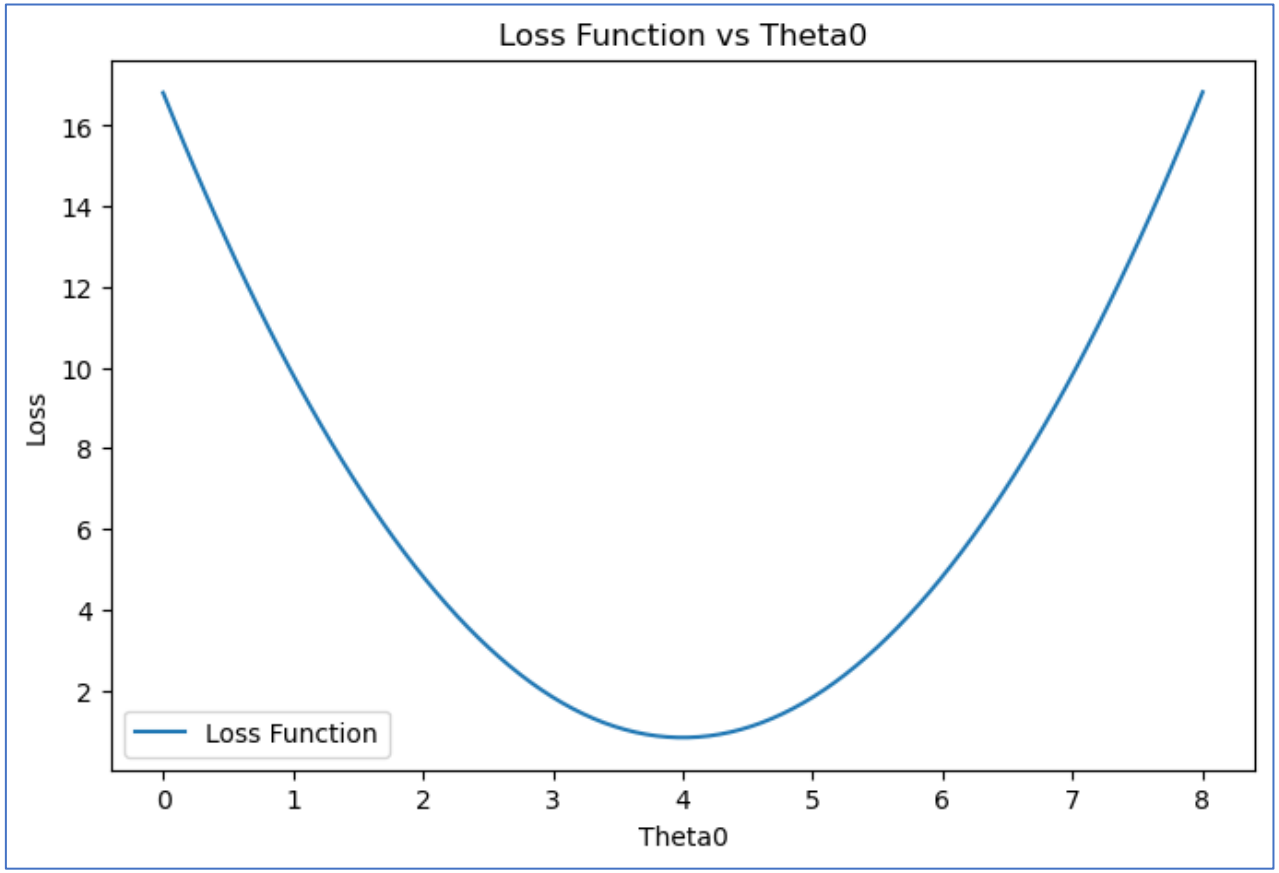
- Gradient Descent Update Rules:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_i$$

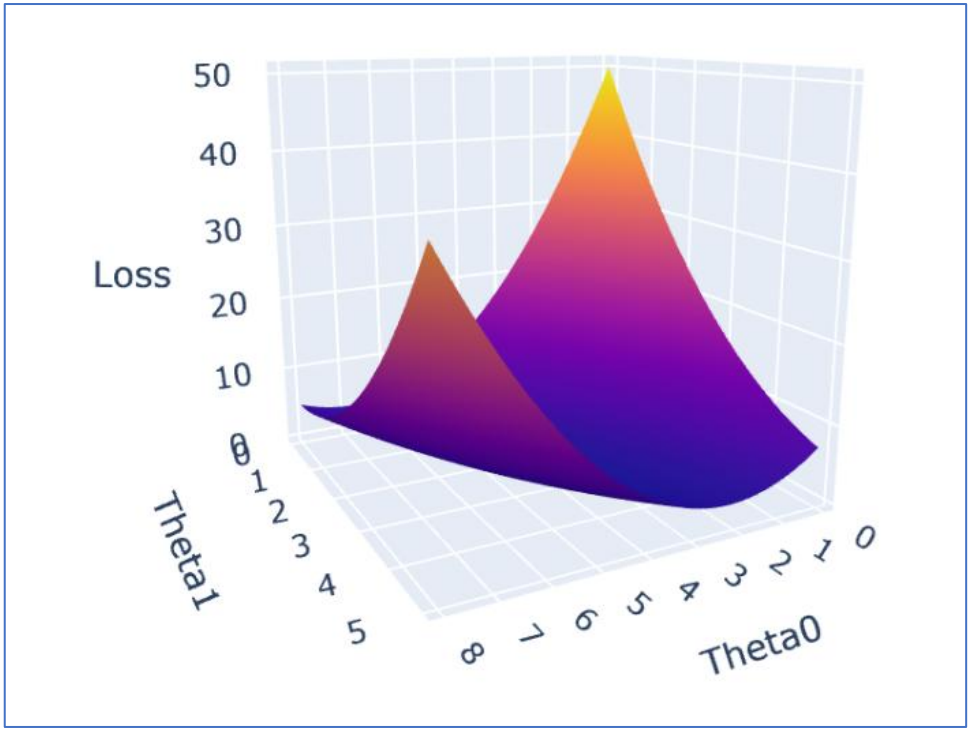
Gradient Descent (Alternative to Normal Equation)

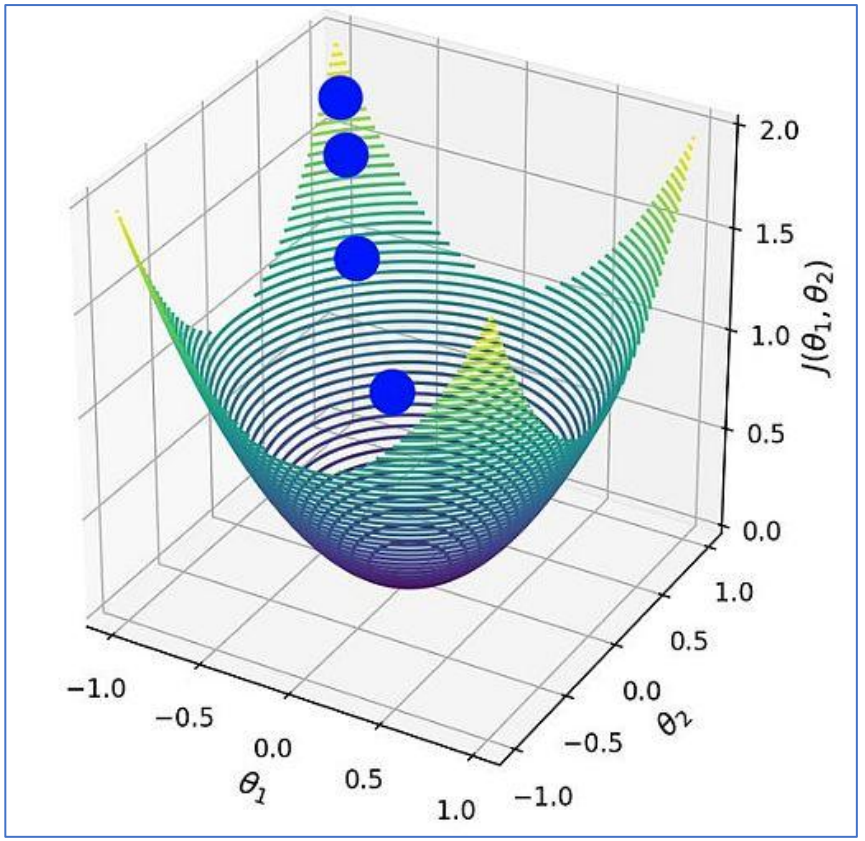
Gradient Descent Algorithm



Gradient Descent (Alternative to Normal Equation)

Gradient Descent Algorithm





Choosing the Learning Rate α in Gradient Descent

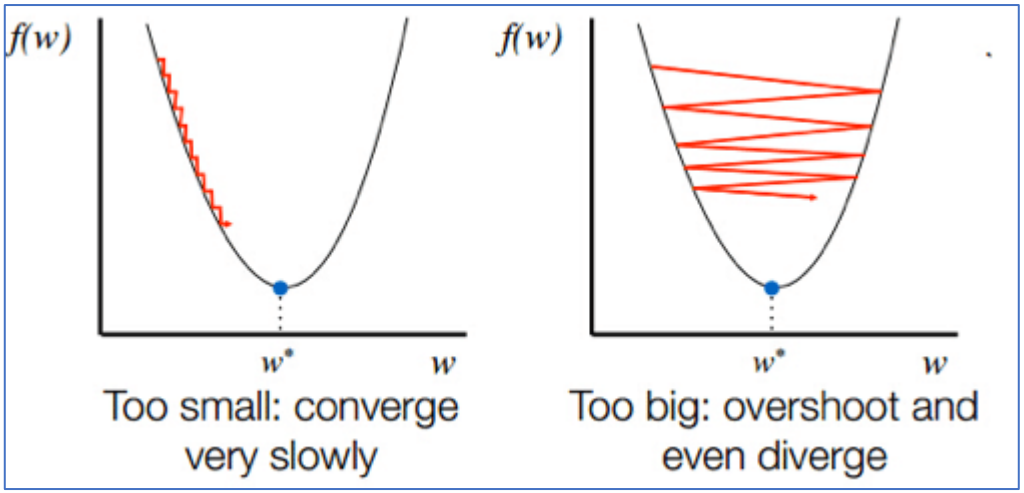
The learning rate α controls how large each step is in the direction of the gradient. Choosing α correctly is crucial for convergence.

Gradient Descent (Alternative to Normal Equation)

Effects of Different Learning Rates

If α is too small

- Convergence is very slow.
- Requires many iterations.
- Computationally expensive.



If α is too large

- Steps are too big, may overshoot the minimum.
- Can cause divergence (cost function increases instead of decreasing).

Optimal α:

- Leads to fast convergence without overshooting.

Guidelines for Choosing α

- Start with a Small Learning Rate
  - A typical starting point is:α=0.01 or 0.1. These values work well for many problems.
- Try Different Values and Observe Convergence (Grid Search)
- Use a Learning Rate Schedule
  - Decrease α over time:
    - where t is the iteration number and k is a small decay factor.

$$\alpha_t = \frac{\alpha_0}{1 + kt}$$

- Use Adaptive Learning Rate Methods : AdaGrad, RMSprop, Adam …

Gradient Descent (Alternative to Normal Equation)

Choosing the number of iteration

It is crucial for ensuring convergence without unnecessary computations. Here are the main strategies:

- Fixed Number of Iterations (Predefined Epochs)
  - Fixed the number of iterations (e.g., 1000, 5000, or 10,000).

    How to choose? Small datasets → 1000 to 5000 iterations

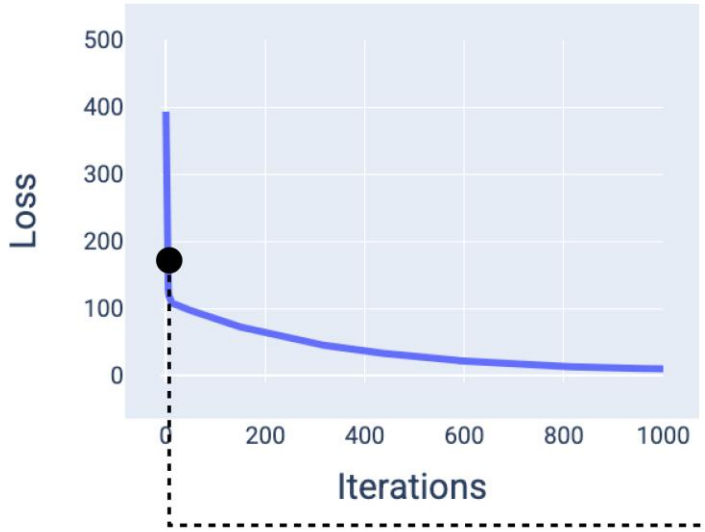
    Large datasets → 10,000+ iterations
- Convergence-Based Stopping Criteria
  - stop when the loss function stops changing significantly

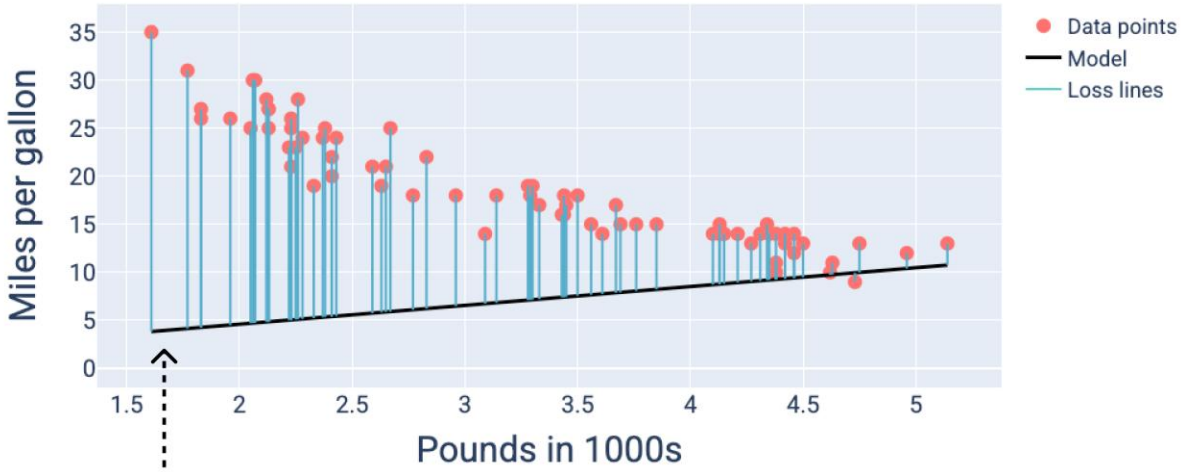
$$\|\nabla L(\theta)\| < \epsilon$$

Gradient Descent (Alternative to Normal Equation)

Choosing the number of iteration

Example (iteration = 4)



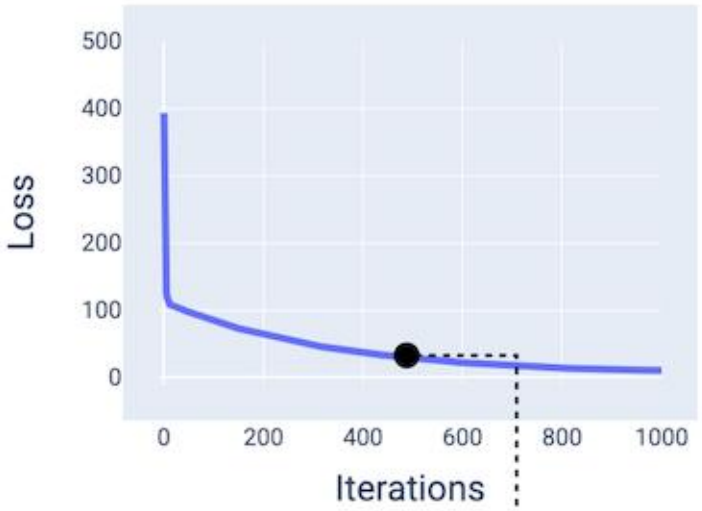


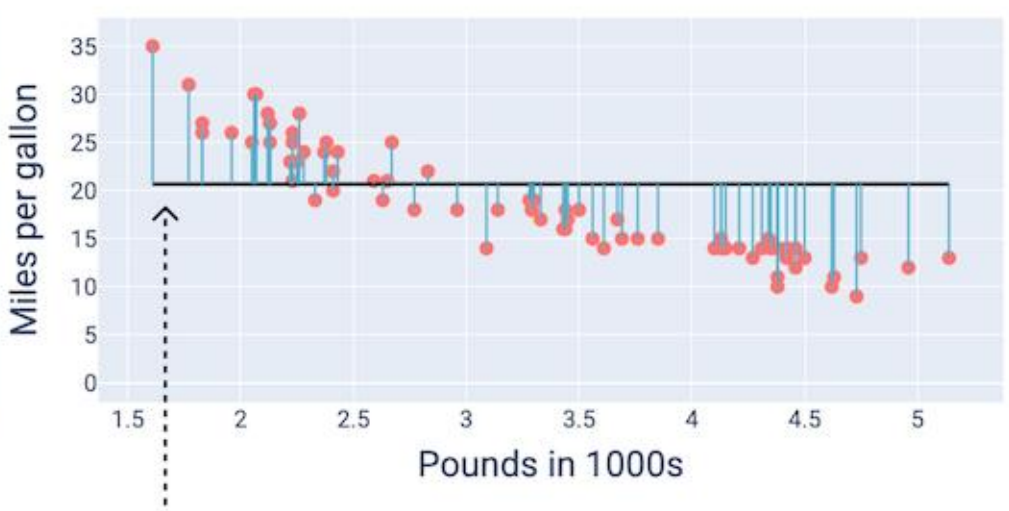
$$\|\nabla L(\theta)\| < \epsilon$$

Gradient Descent (Alternative to Normal Equation)

Choosing the number of iteration

Example (iteration = 400)



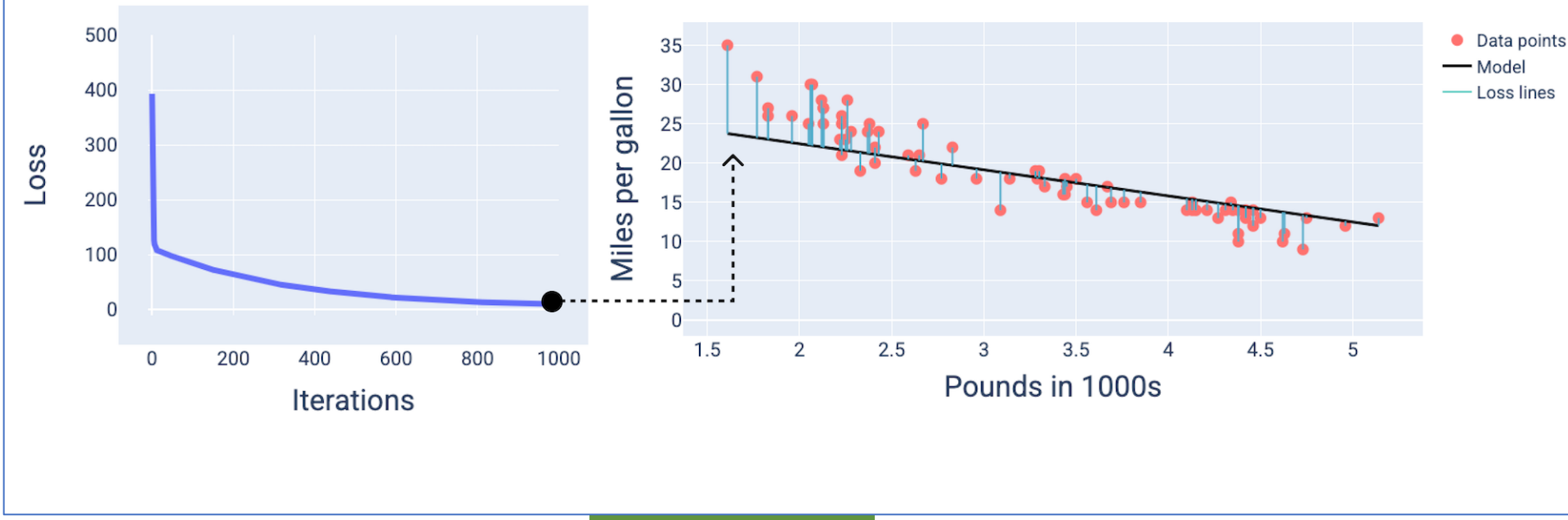


$$\|\nabla L(\theta)\| < \epsilon$$

Gradient Descent (Alternative to Normal Equation)

Choosing the number of iteration

Example (iteration = 1000)



$$\|\nabla L(\theta)\| < \epsilon$$

Gradient Descent (Matrix Calculus)

- For linear regression, the gradient of the cost function is:

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

$$\theta := \theta - \alpha \frac{1}{m} X^T (X\theta - Y)$$

Gradient Descent (Alternative to Normal Equation)

Pros and cons of Gradient Descent

Pros:

- ✓ Scalable to large datasets and high-dimensional problems.
- ✓ Works for a wide range of models (linear, non-linear, deep learning).
- ✓ Can handle non-analytic loss functions (e.g., those with regularization).

Gradient Descent (Alternative to Normal Equation)

Pros and cons of Gradient Descent

Cons:

- × Requires careful tuning of the learning rate.
- × May converge to a local minimum instead of the global minimum.
- × Slower than analytic methods for small datasets.

Gradient Descent (Alternative to Normal Equation)

Variations of Gradient Descent

Different types of gradient descent improve efficiency, stability, and convergence speed.

- Batch Gradient Descent (BGD)
- Stochastic Gradient Descent (SGD)
- Mini-Batch Gradient Descent

Multivariate Linear Regression

Hypothesis Function

From One to Many Features

- In univariate linear regression, we had:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- For multivariate linear regression, where we have multiple input features

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Hypothesis Function

Matrix Form Representation:

- To simplify this equation, we express it using matrices:

$$h_\theta(X) = X\theta$$

- X is the design matrix (including a column of ones for θ_0) and θ is the parameter vector:θ=[θ_0 θ_1 θ_2 … θ_n]:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

Hypothesis Function

Matrix Form Representation:

- To simplify this equation, we express it using matrices:

$$h_\theta(X) = X\theta$$

- X is the design matrix (including a column of ones for θ_0) and θ is the parameter vector:θ=[θ_0 θ_1 θ_2 … θ_n]:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

Cost Function

- The goal of linear regression is to minimize the difference between predicted values and actual values using the Mean Squared Error (MSE).

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

- Matrix Form

$$J(\theta) = \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

Optimization : Least Squares Method

Normal Equation (Direct Solution)

- we can find the optimal parameters directly using the Normal Equation

$$\theta_j = \frac{\sum_{i=1}^{m}(x_j^{(i)} - \bar{x}_j)(y^{(i)} - \bar{y})}{\sum_{i=1}^{m}(x_j^{(i)} - \bar{x}_j)^2}, \quad \text{for } j = 1, 2, ..., n$$

$$\theta_0 = \bar{y} - \theta_1\bar{x}_1 - \theta_2\bar{x}_2 - ... - \theta_n\bar{x}_n$$

- **Matrix Form**

$$\theta = (X^T X)^{-1} X^T Y$$

Optimization : Gradient Descent

Gradient Descent

- To minimize $J(\theta)$, we use Gradient Descent, which updates θ iteratively: **(for j: 0 .. n)**

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

Matrix Form By writing this in matrix notation, we get the compact update rule:

$$\theta := \theta - \alpha \frac{1}{m} X^T (X\theta - Y)$$

Polynomial Linear Regression

Polynomial Linear Regression

Hypothesis Function

- Polynomial regression is an extension of linear regression where we introduce higher-degree terms of the input features. The hypothesis function is:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_d x^d$$

where:

- d is the degree of the polynomial.
- θ are the parameters (coefficients) to learn.
- x is the input feature.

Polynomial Linear Regression

Hypothesis Function

Matrix Formulation

- If we define a design matrix X where each column corresponds to x^j for $j=0,1,...,dj$, then:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^d \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

This allows us to write the hypothesis in vector form: d is the degree of the polynomial.

$$h_\theta(X) = X\theta$$

Polynomial Linear Regression

Cost Function

- Like in linear regression, we use Mean Squared Error (MSE) as the cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Matrix Formulation

- Using the matrix notation:

$$J(\theta) = \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

where:

- X is the polynomial feature matrix.
- Y is the target vector.

Optimization Algorithms

Gradient Descent (Iterative Solution)

- We update the parameters iteratively using gradient descent:

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

- To update θ, we take the partial derivative of $J(\theta)$ with respect to θj : (for each j=0,1,…,d)

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) (x^{(i)})^j$$

Gradient Descent (Iterative Solution)

Matrix Formulation

- Gradient Computation :

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T (X\theta - Y)$$

- Thus, the update rule in vector form:

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

Polynomial Linear Regression

Feature Scaling (Important for Gradient Descent!)

- Since polynomial features can lead to very large values (e.g., xd grows rapidly), feature scaling (e.g., standardization) helps gradient descent converge faster.

$$x_j = \frac{x_j - \mu_j}{\sigma_j}$$

where :

- μ_j is the mean of feature x_j.
- σ_j is the standard deviation.

Regularization of Ridge and Lasso

Regularization of Ridge and Lasso

Motivation : Why Do We Need Regularization?

Regularization is a technique to prevent overfitting by adding a penalty term to the cost function

The Problem of Overfitting

- A regression model with too many features or high-degree polynomial terms can memorize noise instead of capturing the true pattern.
- This leads to low training error but high test error, meaning poor generalization.
- Regularization helps prevent this by penalizing large coefficients and reducing model complexity.

Motivation for Regularization

- Example

Regularization

- When we penalize the weights θ_3 and θ_4 and make them too small, very close to zero. It makes those terms negligible and helps simplify the model.

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$

Regularization of Ridge and Lasso

Motivation for Regularization

- Example

Regularization

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

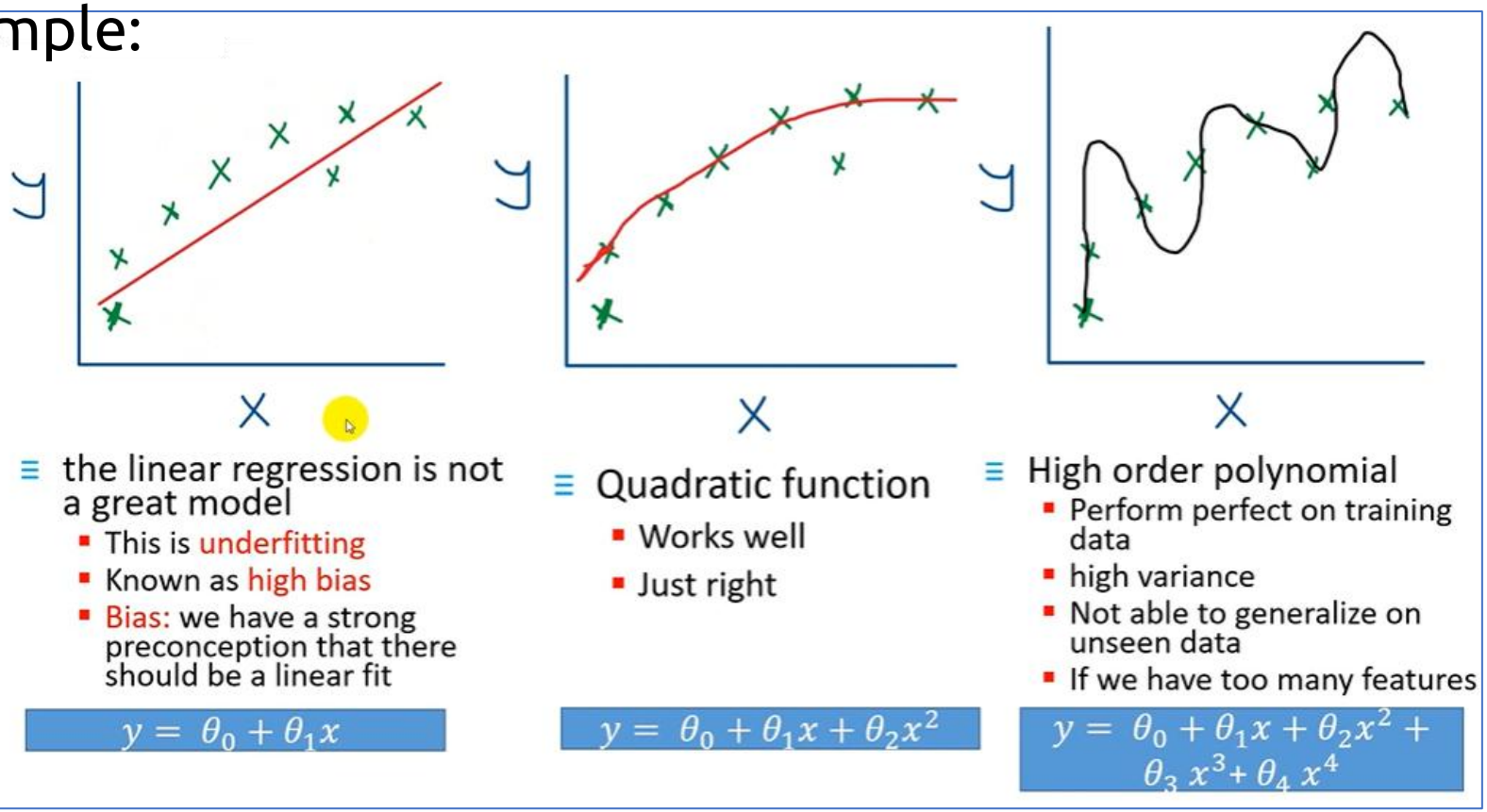
$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$

Motivation for Regularization

Why Do We Need Regularization?

Example:



≡ the linear regression is not a great model
- This is underfitting
- Known as high bias
- Bias: we have a strong preconception that there should be a linear fit

$$y = \theta_0 + \theta_1 x$$

≡ Quadratic function
- Works well
- Just right

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

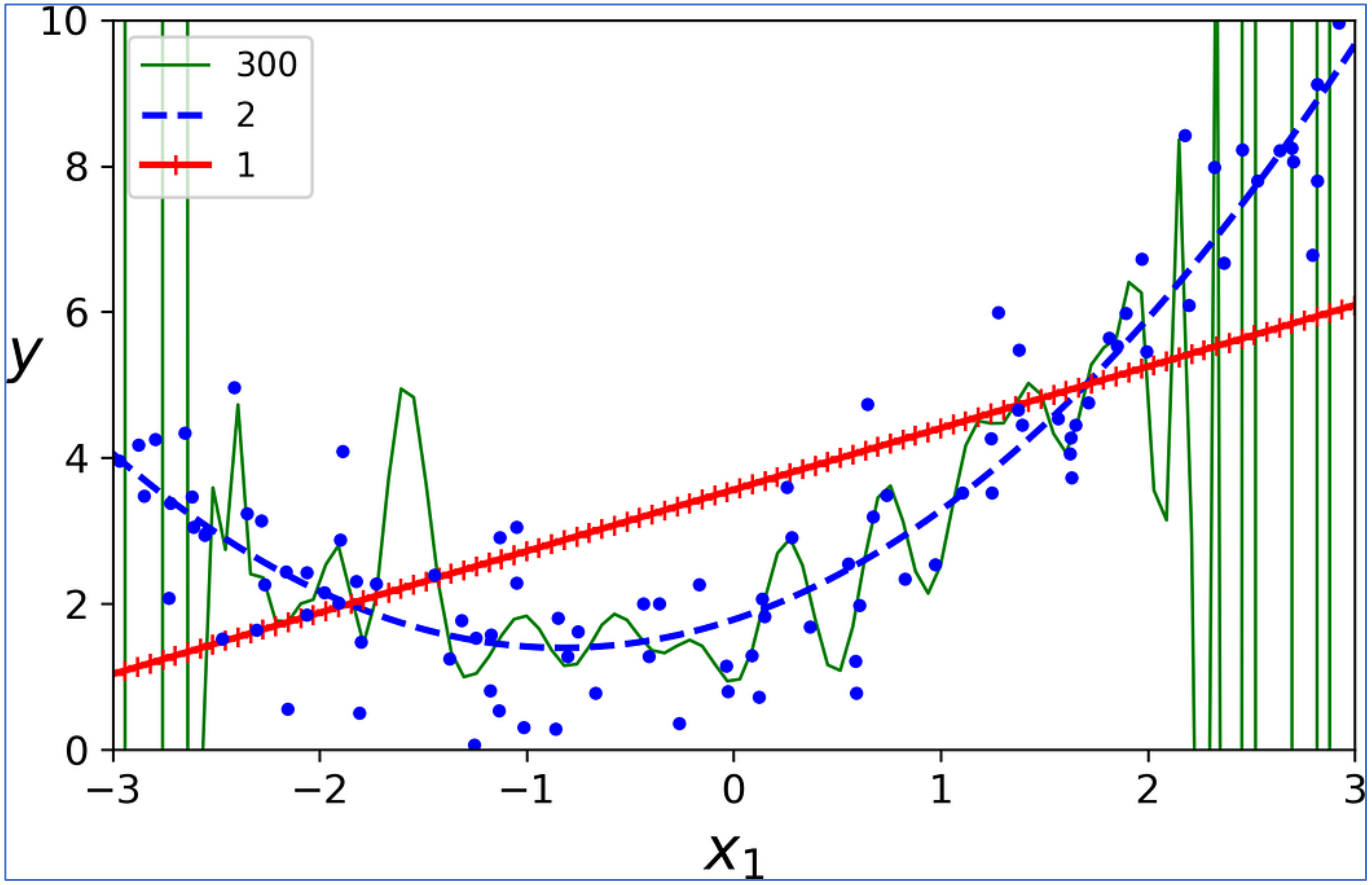
≡ High order polynomial
- Perform perfect on training data
- high variance
- Not able to generalize on unseen data
- If we have too many features

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Motivation for Regularization

Why Do We Need Regularization?

Example:



Motivation for Regularization

Why Do We Need Regularization?

Bias-Variance Tradeoff

- Regularization balances two opposing forces:
  - **Bias (underfitting):** Model is too simple to capture patterns.
  - **Variance (overfitting**): Model fits training data too well but fails on new data.
- **Goal → Find an optimal middle ground.**

Regularization of Ridge and Lasso

Motivation for Regularization

To prevent overfitting, we use Regularization:

- Ridge Regression (L2 Regularization)
- Lasso Regression (L1 Regularization)

Motivation for Regularization

Ridge Regression (L2 Regularization)

- Adds a penalty on the sum of squared coefficients.
- Shrinks coefficients toward zero but never sets them exactly to zero.
- Helps reduce multicollinearity and stabilize the model.

Motivation for Regularization

Ridge Regression (L2 Regularization)

- Cost Function:

$$J(\theta) = \frac{1}{m} \sum (y_i - h_\theta(X_i))^2 + \lambda \sum \theta^2$$

where:

$$h_\theta(X) = X\theta$$

Motivation for Regularization

Ridge Regression (L2 Regularization)

- Cost Function:

$$J(\theta) = \frac{1}{m}\sum(y_i - h_\theta(X_i))^2 + \lambda\sum\theta^2$$

where:

where λ (regularization parameter) controls the penalty:

- If $\lambda = 0 \rightarrow$ Equivalent to standard linear regression.
- If λ **is large** $\rightarrow$ More shrinkage, leading to **simpler models**.

Motivation for Regularization

Lasso Regression (L1 Regularization)

- Adds a penalty on the sum of absolute values of coefficients.
- Can force some coefficients to be exactly zero, effectively performing feature selection.
- So:
  - It removes irrelevant features (Automatic feature selection.).
  - Unlike Ridge, it doesn't shrink coefficients smoothly.

Motivation for Regularization

Lasso Regression (L1 Regularization)

- Cost Function:

$$J(\theta) = \frac{1}{m} \sum (y_i - h_\theta(X_i))^2 + \lambda \sum |\theta|$$

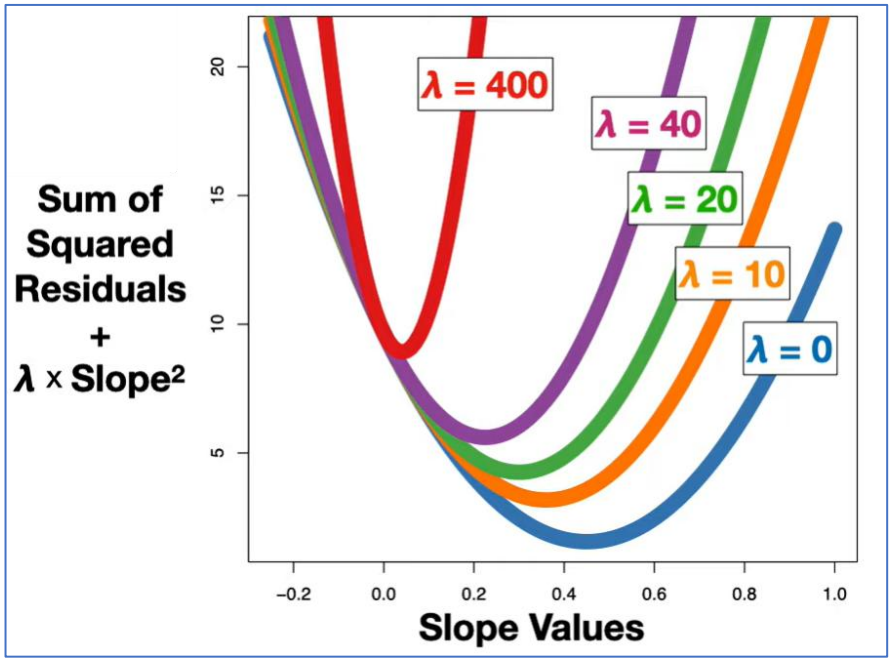
Motivation for Regularization

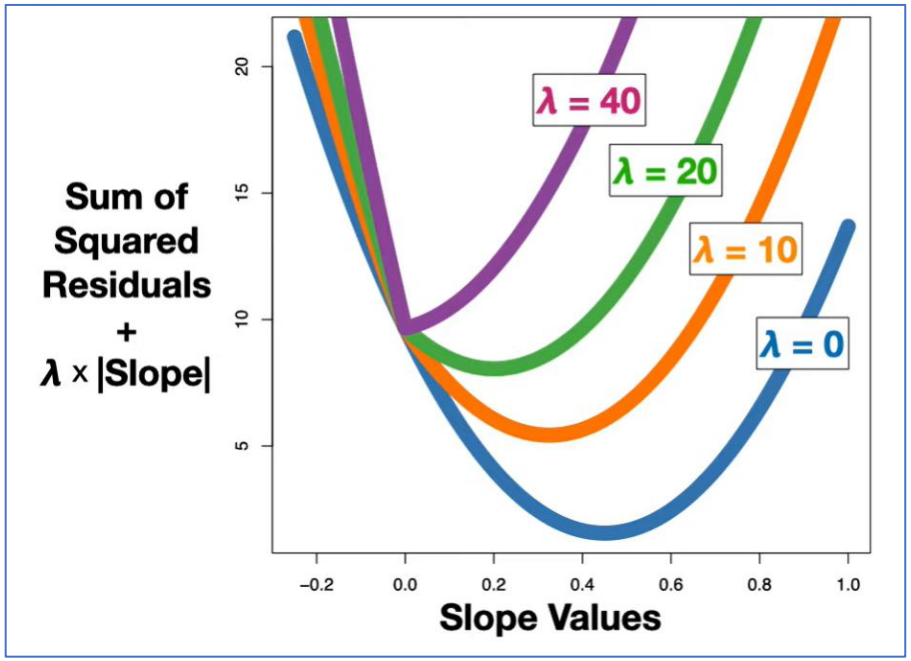
How to Choose λ in Regularization?

- The choice of λ in Ridge and Lasso regression is crucial because it controls the trade-off between bias and variance.
- $\lambda=0 \rightarrow$ No regularization, equivalent to standard linear regression.
- Small λ values $\rightarrow$ Less penalty, model is close to ordinary least squares (OLS).
- Large λ values $\rightarrow$ Higher penalty, reducing the magnitude of coefficients.
- Too large $\lambda \rightarrow$ Can underfit the data (parameters too small)

Motivation for Regularization

Example





Methods to Select λ

1. Cross-Validation (Best Approach)

- Use K-Fold Cross-Validation to test different λ values and choose the one that minimizes the validation error.

2. Others:

- Grid Search with Cross-Validation
- Heuristic Rules (Quick Approximation)
- …

Metrics for Regression Model Evaluation

Metrics for Regression Model Evaluation

Mean Absolute Error

Mean Absolute Error (MAE)

$$MAE = \frac{1}{m} \sum_{i=1}^{m} |h_\theta(x_i) - y_i|$$

- Interpretation: Measures the average absolute difference between actual and predicted values.
- Pros: Easy to understand, less sensitive to outliers than MSE.
- Cons: Does not penalize large errors as much as MSE.

Metrics for Regression Model Evaluation

Mean Squared Error (MSE)

Mean Squared Error (MSE)

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

- Interpretation: Measures the average squared difference between actual and predicted values.
- Pros: Penalizes larger errors more than MAE (useful when large errors are unacceptable).
- Cons: Sensitive to outliers.

Metrics for Regression Model Evaluation

Mean Squared Error (MSE)

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2}$$

- Interpretation: Similar to MSE but in the same unit as the target variable.
- Pros: More interpretable than MSE because it has the same units as y.
- Cons: Still sensitive to outliers.

Metrics for Regression Model Evaluation

Mean Squared Error (MSE)

R-squared (R^2) Score

$$R^2 = 1 - \frac{\sum(h_\theta(x_i) - y_i)^2}{\sum(y_i - \bar{y})^2}$$

- Interpretation: Represents the proportion of variance in y explained by the model.
- $R^2 = 1 \rightarrow$ Perfect model.
- $R^2 = 0 \rightarrow$ Model does no better than mean prediction.
- $R^2 < 0 \rightarrow$ Model is worse than using the mean of y.

Metrics for Regression Model Evaluation

Mean Squared Error (MSE)

Others:

- Adjusted R-squared.
- Mean Absolute Percentage Error (MAPE)
- …

Thank you for your attention…