solidchart-line solidcode soliddatabase

Contents

Introduction to Linear Regression

solidstar Why This Algorithm Matters

Simple Linear Regression is a **super important algorithm** because the techniques that we are going to learn in this will also be applicable in **deep learning**. When you're learning the first neural network, which is called an **Artificial Neural Network (ANN)**, the same foundational concepts apply.

Problem Statement

solidquestion-circle What Problems Does Linear Regression Solve?

As the name suggests — **Regression**. In supervised machine learning technique, if you have a **regression problem statement**, we can solve it with the help of simple linear regression.

Understanding Through Example

Let's consider a specific and easy dataset:

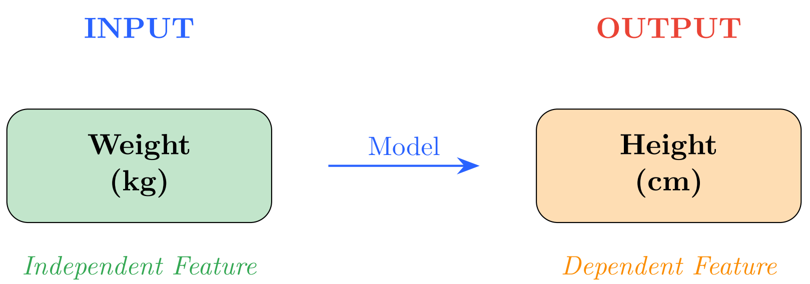
Weight (kg)	Height (cm)
74	170
80	180
75	175.5

Table 1: Sample Dataset for Linear Regression

solidbullseye Main Aim

Our main aim is that we need to **train a model**. This specific model, whenever we give a **new weight**, should be able to **predict the height**.

Feature Types



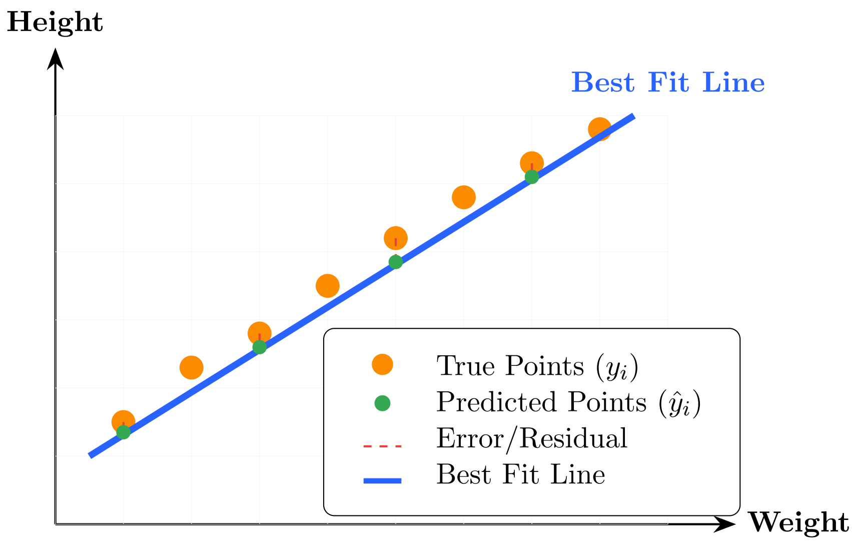
Simple vs Multiple Linear Regression

solidinfo-circle Key Distinction

- **Simple Linear Regression:** When we have just **ONE input feature** and one output feature
- **Multiple Linear Regression:** When we have **multiple input features**

It is very important to understand simple linear regression first. Then automatically whatever terminologies and maths you learn here will get applied to multiple linear regression also.

Geometric Interpretation: The Best Fit Line



solidtarget The Core Idea

With the help of simple linear regression, we try to create a **best fit line**. This best fit line will help us to do the prediction for the new weight.
How does prediction happen?

1. Once we get the best fit line
2. When we get a new data point (new weight)
3. We project it onto the best fit line
4. Read the corresponding y-value (height)

The best fit line should be created such that the **summation of all distances** (errors) between true points and predicted points should be **minimal**.

Mathematical Notation & Fundamentals

solidexclamation-circle Important Note

Before we understand the maths behind it, we really need to understand some of the notation that will be used while explaining this entire machine learning algorithm.

Equation of the Best Fit Line

solidedit Different Notations for the Same Equation

The equation of a straight line can be written in multiple ways:

Notation	**Form**
Standard Form	$y = mx + c$
Research Paper Style	$y = \beta_0 + \beta_1 x$
Andrew Ng's Notation	$\boxed{h_\theta(x) = \theta_0 + \theta_1 x}$

We will use Andrew Ng's notation: $h_\theta(x) = \theta_0 + \theta_1 x$

Understanding Each Component

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

$h_\theta(x)$	θ_0	θ_1	x
Predicted Value (Hypothesis)	Intercept (Y-intercept)	Slope (Coefficient)	Independent Feature (Input)

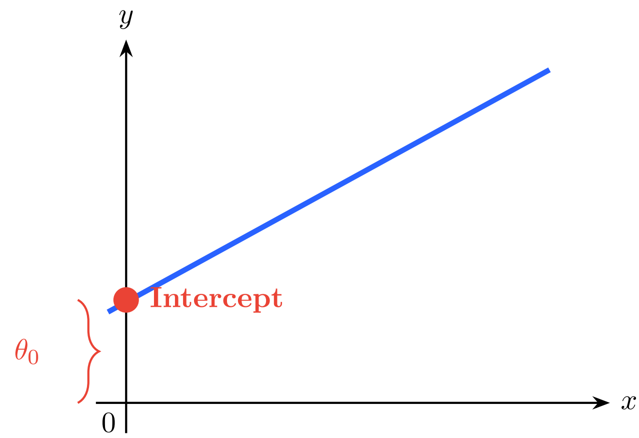
Understanding θ_0 (Intercept)

solidmap-marker-alt The Intercept θ_0

θ_0 is the Intercept
If my x value is zero, then:

$$h_\theta(x) = \theta_0 + \theta_1 \cdot 0 = \theta_0$$

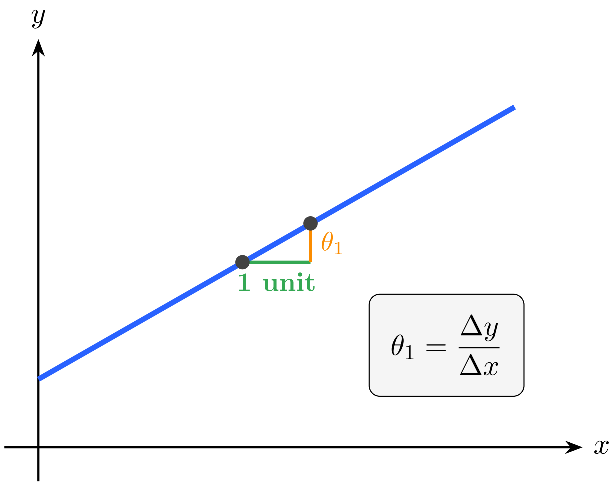
This indicates: **When x is zero, where does the line meet the y-axis?**
The point where the line intersects the y-axis is called the **intercept**.



Understanding θ_1 (Slope/Coefficient)

solidchart-line The Slope θ_1

θ_1 is the Slope or Coefficient
It indicates: **With unit movement in the x-axis, what is the movement with respect to the y-axis?**
This is called the **slope movement**.



Extension to Multiple Features

solidlayer-group Multiple Independent Features

If we have many independent features, the equation becomes:

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n}$$

- θ_0 — Always ONE intercept
- $\theta_1, \theta_2, \ldots, \theta_n$ — Slopes for each feature
- $x_1, x_2, \ldots, x_n$ — Independent features

Since we have ONE independent feature in simple linear regression, we have just ONE slope (θ_1).

Predicted Points Notation

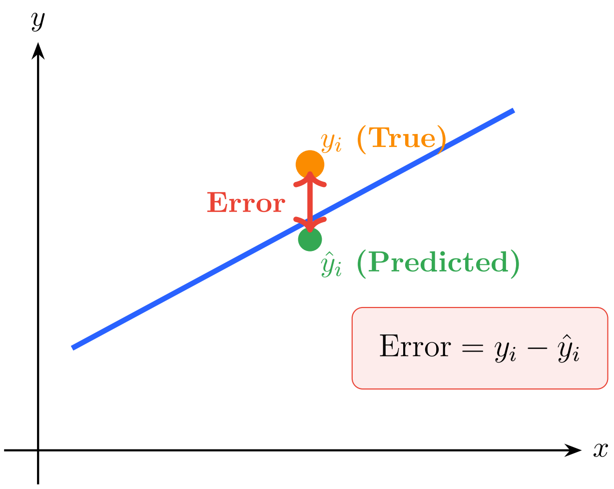
solidhat-wizard The $\hat{y}$ Notation

The predicted points are denoted by:

$$\hat{y} = h_\theta(x)$$

Where $\hat{y}$ ("y-hat") represents the **predicted points** obtained from our model.

Understanding Error



solidbalance-scale Error Definition

$$\boxed{\text{Error} = y - \hat{y}}$$

Error is the difference between:

- y — The true point (actual output from data)

- $\hat{y}$ — The predicted point (from best fit line)

solidbullseye Main Aim Restated

Our main aim is to create a **best fit line** wherein when we try to calculate or do the **summation of all errors**, it should be **minimal**.
If it is not minimal and there is another best fit line that minimizes this error, we select that specific best fit line instead.

Cost Function: Mean Squared Error

The Need for a Cost Function

solidexclamation-circle The Problem

We cannot just randomly create many best fit lines and check which one has the least error. This is **not efficient**.
We need to find an **optimized way**:

1. Create one best fit line
2. Rotate this line by changing values of intercept (θ_0) and coefficient (θ_1)
3. Find the best fit line with **minimal error**

The Cost Function Formula

solidfunction Mean Squared Error (MSE) Cost Function

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

Where:

- $J(\theta_0, \theta_1)$ — Cost Function
- m — Number of data points
- $h_\theta(x^{(i)})$ — Predicted points
- $y^{(i)}$ — True points (actual output)
- The squaring makes this the "Mean **Squared** Error"

solidquestion-circle Why Squared?

The reason we are **squaring** is because the technique/cost function we are using is called **Mean Squared Error (MSE)**. There are advantages of using MSE which we will discuss.
Other cost functions exist:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

Our Final Objective

solidflag-checkered What We Need to Solve

MINIMIZE $J(\theta_0, \theta_1)$

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

How? By continuously changing θ_0 and θ_1 values to find the best fit line where the summation of error is **minimal as possible**.

Gradient Descent: Finding the Best Fit Line

Simplified Example

To understand gradient descent in 2D, let's assume $\theta_0 = 0$ (line passes through origin).

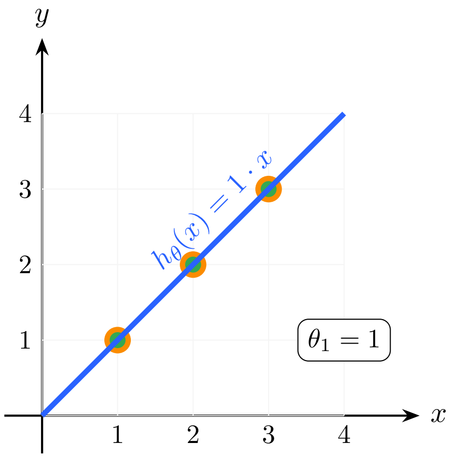
solidddatabase Sample Dataset

Consider the dataset:

x	y
1	1
2	2
3	3

With $\theta_0 = 0$, our equation becomes: $h_\theta(x) = \theta_1 \cdot x$

Case 1: $\theta_1 = 1$



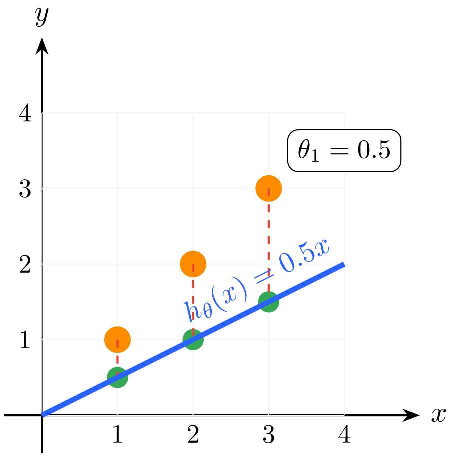
solidsquare-root-alt Important Formula

Computing Cost Function when $\theta_1 = 1$:

$$J(\theta_1) = \frac{1}{2 \cdot 3}\left[(1-1)^2 + (2-2)^2 + (3-3)^2\right] \tag{2}$$
$$= \frac{1}{6}[0 + 0 + 0] \tag{3}$$
$$= \boxed{0} \tag{4}$$

Result: Cost function is **ZERO**! The line passes through ALL points perfectly.

Case 2: $\theta_1 = 0.5$



solidsquare-root-alt Important Formula

Computing Cost Function when $\theta_1 = 0.5$:

$$J(\theta_1) = \frac{1}{2 \cdot 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \tag{5}$$
$$= \frac{1}{6}[0.25 + 1 + 2.25] \tag{6}$$
$$= \frac{3.5}{6} \approx \boxed{0.58} \tag{7}$$

Case 3: $\theta_1 = 0$

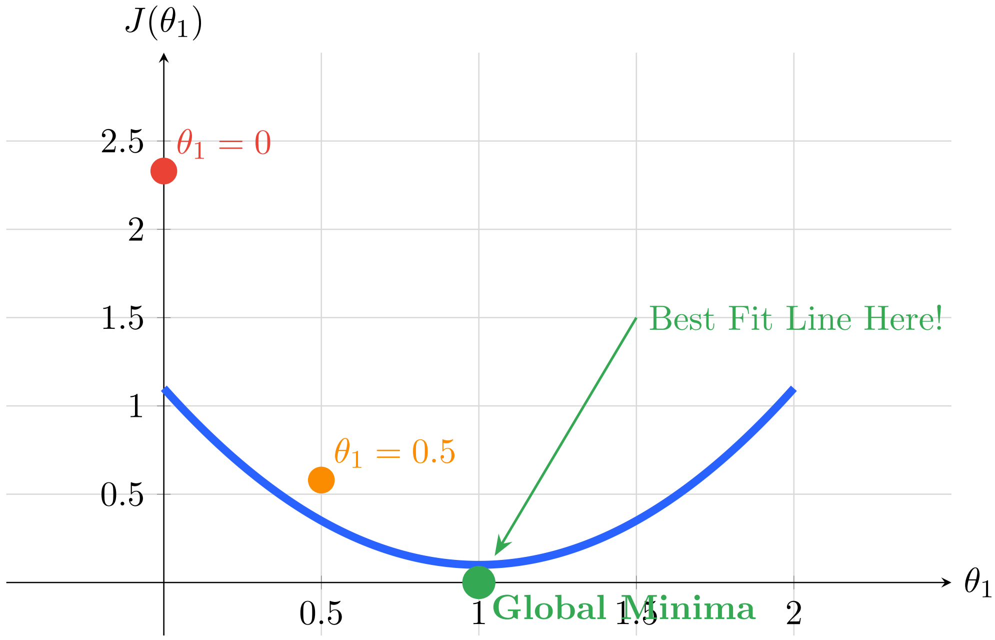
solidsquare-root-alt Important Formula

Computing Cost Function when $\theta_1 = 0$:

$$J(\theta_1) = \frac{1}{2 \cdot 3}\left[(0-1)^2 + (0-2)^2 + (0-3)^2\right] \tag{8}$$
$$= \frac{1}{6}[1 + 4 + 9] \tag{9}$$
$$= \frac{14}{6} \approx \boxed{2.33} \tag{10}$$

This is NOT a good fit! Error is quite high.

The Gradient Descent Curve



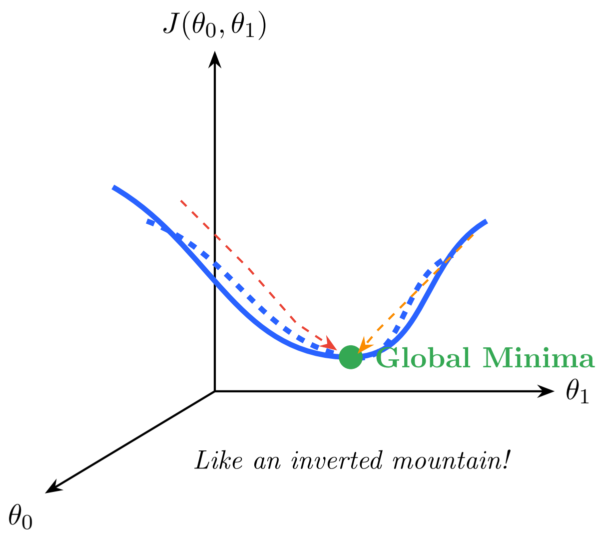
solidmountain The Gradient Descent Curve

This entire curve is called the **Gradient Descent** curve.

- The lowest point is called the **Global Minima**
- At the global minima, our cost function is minimal
- This is where we get our **best fit line**
- We need to reach this point by changing θ_0 and θ_1 values

Key Insight: Gradient descent is super important even in deep learning techniques!

3D Visualization (When $\theta_0 \neq 0$)



Like an inverted mountain!

solidexclamation-triangle Important Note

When we have both θ_0 and θ_1, the gradient descent becomes a **3D surface** (like an inverted mountain or bowl). Both parameters must converge to their optimal values to reach the global minima.

The Convergence Algorithm

The Problem with Random Selection

solidtimes-circle Inefficient Approach

When we were finding our cost function, we were **randomly changing** our θ_1 value (slope):

- First time: $\theta_1 = 1$
- Second time: $\theta_1 = 0.5$
- Third time: $\theta_1 = 0$

This is **NOT an efficient technique**! We need an algorithm that:

1. Initializes one θ_1 value
2. Automatically increases or decreases based on the gradient descent

The Convergence Algorithm

solidrepeat The Convergence Algorithm

Repeat until convergence:

$$\boxed{\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)} \tag{11}$$

Where:

- θ_j — The parameter being updated
- α — Learning rate (small positive number)
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ — Derivative (slope) of cost function
- $:=$ — Assignment operator (iterative update)

"Until convergence" means until we reach the global minima point.

Understanding the Derivative (Slope)

solidcalculator What Does the Derivative Tell Us?

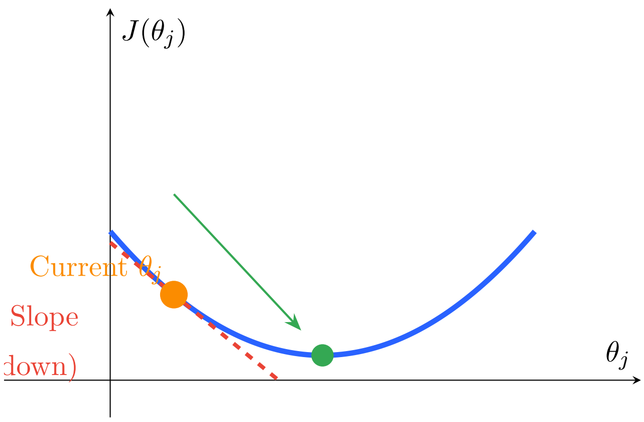
The derivative $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ calculates the **slope** at the current point on the gradient descent curve.
We create a **tangent line** at the current point and determine:

- Is it a **positive slope** or **negative slope**?

How to identify: Look at the **right side** of the tangent line:

- If pointing **upward** $\rightarrow$ Positive slope
- If pointing **downward** $\rightarrow$ Negative slope

Case 1: Negative Slope (Left of Minima)



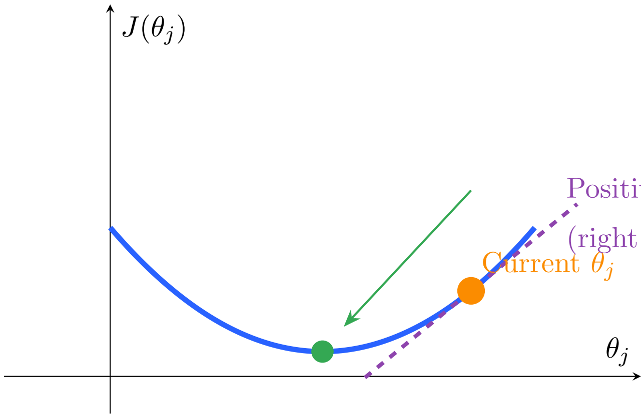
solidarrow-right When Slope is Negative

$$\theta_j := \theta_j - \alpha \cdot (\text{negative value})$$

$$\theta_j := \theta_j + (\text{positive value})$$

Result: θ_j **INCREASES** $\rightarrow$ Moves toward global minima!

Case 2: Positive Slope (Right of Minima)



solidarrow-left When Slope is Positive

$$\theta_j := \theta_j - \alpha \cdot (\text{positive value})$$

$$\theta_j := \theta_j - (\text{positive value})$$

Result: θ_j **DECREASES** $\rightarrow$ Moves toward global minima!

The Learning Rate (α)

solidtachometer-alt Learning Rate α

Learning Rate (α) is a smaller value that controls the **speed of convergence**.

- Typically: $\alpha = 0.001$ (common practice)
- In sklearn library: $\alpha = 0.001$ is the default for simple linear regression

α too small

Takes more time to converge

solidhourglass-half

α just right

Optimal convergence

solidcheck-circle

α too big

May never converge (jumps around)

solidtimes-circle

soliduser-tie Interview Question

Q: What is the importance of learning rate?
A: The learning rate controls the convergence rate — how slow or fast the convergence should happen. A very small value takes more time; a very big value may cause the algorithm to never converge (continuously jumping).

Complete Convergence Equations

The Full Update Equations

solidcode Final Convergence Algorithm

Repeat until convergence:

Update θ_0 (Intercept):

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

Update θ_1 (Slope):

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

Where:

- α = Learning rate
- m = Number of data points
- $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
- $y^{(i)}$ = Actual output

Derivation Summary

solidgraduation-cap How We Got These Equations

Starting from:

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

For $j = 0$ (intercept):

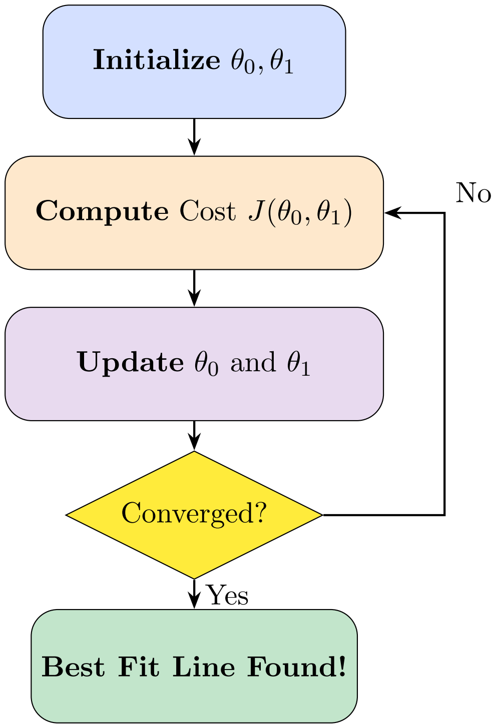
$$\frac{\partial}{\partial \theta_0} J = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot 1$$

The derivative of $(\theta_0 + \theta_1 x)$ w.r.t. θ_0 is 1.

For $j = 1$ (slope):

$$\frac{\partial}{\partial \theta_1} J = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

The derivative of $(\theta_0 + \theta_1 x)$ w.r.t. θ_1 is x.


Initialize θ_0, θ_1
Compute Cost $J(\theta_0, \theta_1)$
No
Update θ_0 and θ_1
Converged?
Yes
Best Fit Line Found!

Multiple Linear Regression

Simple vs Multiple Linear Regression

Simple Linear Regression	→	Multiple Linear Regression
ONE input feature		**MANY** input features
$h_\theta(x) = \theta_0 + \theta_1 x$		$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$
Example: Weight → Height		Example: Rooms, Size, Location → Price

House Pricing Example

solidhome House Pricing Dataset

No. of Rooms	Size of House	Location	Price
x_1	x_2	x_3	y
Independent Features			Dependent

Here we have **3 input features**, so this is **Multiple Linear Regression**.

The Extended Equation

solidsuperscript Multiple Linear Regression Equation

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3}$$

Term	Meaning	Example
θ_0	Intercept (always ONE)	Y-intercept
θ_1	Coefficient for x_1	Slope for "Number of Rooms"
θ_2	Coefficient for x_2	Slope for "Size of House"
θ_3	Coefficient for x_3	Slope for "Location"

Key Point: As the number of input features increases, that many coefficients will increase. But θ_0 (intercept) is always **ONE**.

Gradient Descent in Higher Dimensions

solidcubes Higher Dimensional Gradient Descent

With multiple features:

- The gradient descent surface becomes multi-dimensional
- Cannot draw 4D+ diagrams, but imagine an **inverted mountain**
- All coefficients $(\theta_0, \theta_1, \theta_2, \theta_3, \ldots)$ must converge to the global minima

The cost function becomes:

$$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Performance Metrics: R^2 and Adjusted R^2

R-Squared (R^2)

solidpercentage R-Squared Formula

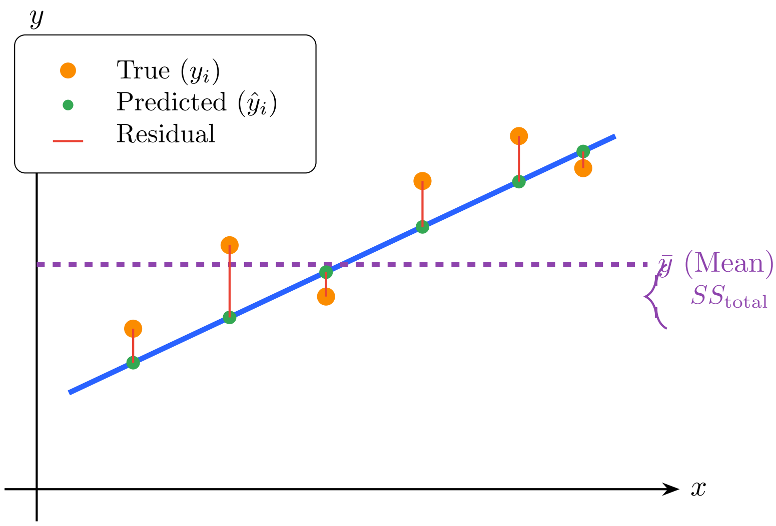
$$\boxed{R^2 = 1 - \frac{SS_{\text{residual}}}{SS_{\text{total}}}}$$

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

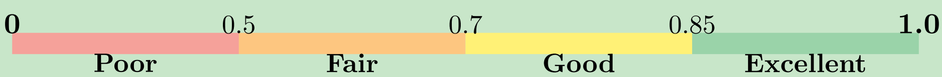
Where:

- SS_{residual} = Sum of Squared Residuals (errors)
- SS_{total} = Sum of Squared Total (from mean)
- y_i = True values
- $\hat{y}_i$ = Predicted values
- $\bar{y}$ = Mean of true values

Visual Understanding of R^2



solidchart-line Interpreting R^2



- $R^2 = 0.70$ means 70% accuracy

- $R^2 = 0.85$ means 85% accuracy
- $R^2 = 0.90$ means 90% accuracy
- **Higher R^2 (closer to 1) → More accurate model**

The Problem with R^2

solidbug Problem with R-Squared

Even if you add a feature that is **NOT correlated** with the output feature, the R^2 value will still **increase**!
Example:

Features Used	R^2 **Value**
Size of House	0.75
Size + No. of Bedrooms	0.80
Size + Bedrooms + Location	0.85
Size + Bedrooms + Location + Gender (irrelevant!)	0.87

Gender has NO correlation with price, yet R^2 still increased!

Adjusted R-Squared

solidbalance-scale Adjusted R-Squared Formula

$$\boxed{R^2_{\text{adjusted}} = 1 - (1 - R^2) \cdot \frac{n-1}{n-p-1}}$$

Where:

- n = Number of data points
- p = Number of independent features
- R^2 = Original R-squared value

solidmagic How Adjusted R^2 Works

Adjusted R^2 **penalizes** for adding features that are **not correlated** with the output.

Scenario	R^2	**Adjusted** R^2
2 correlated features	90%	86%
Add 1 correlated feature	92%	88% (increases)
Add 1 **uncorrelated** feature	93%	83% (decreases!)

Key Insight:

- If new feature is correlated → Adjusted R^2 **increases**

- If new feature is NOT correlated $\rightarrow$ Adjusted R^2 **decreases**

Complete Summary

solidclipboard-check Summary

Key Concepts Covered:

1. **Simple Linear Regression:** ONE input feature, creates best fit line
2. **Equation:** $h_\theta(x) = \theta_0 + \theta_1 x$
    - θ_0 = Intercept (where line meets y-axis)
    - θ_1 = Slope (rate of change)
3. **Cost Function (MSE):**

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

4. **Gradient Descent:** Bowl-shaped curve; aim is to reach **Global Minima**
5. **Convergence Algorithm:**

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

6. **Learning Rate (α):** Controls speed of convergence (typically 0.001)
7. **Multiple Linear Regression:** Multiple input features

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

8. **Performance Metrics:**
    - R^2 = Coefficient of determination (can be misleading)
    - Adjusted R^2 = Penalizes uncorrelated features (more reliable)

solidcheck-circle **These techniques are foundational for Deep Learning & Neural Networks!**

End of Linear Regression Notes

solidbrain Foundation for Machine Learning & Deep Learning

solidrocket Ready for Practical Implementation!