Machine Learning

Linear Regression with one variable

By :
Saja Malek Hassan
Maryam Mohamed Ragab
Shahd Ashraf Ramadan

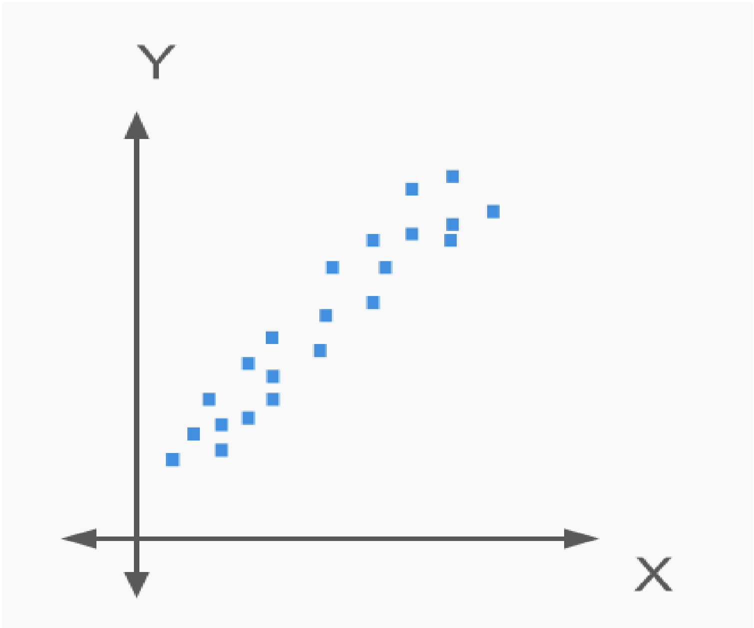
Agenda

- Introduction
- Types of Regression
- Problem Statement
- Hypothesis Function
- Cost Function
- Gradient Descent
- Evalution
- Applications
- Code & Dataset
- Conclusion

Introduction

What is regression?
- Supervised learning algorithm that works with **labeled data** .
- Simple model for predicting continuous values .
- It finds the relationship between input features and output.

Linear Regression is a fundamental statistical method used to define **linear relationship** between a dependent and one or more independent variables. It helps predict **outcomes** by fitting a **straight line** to observed data points, making it easy to interpret and apply.



Examples :
- House price from its size.
- salary from years of experience.
- Price of car from Car's attribute.

Types of Regression

Linear Regression

- **Simple Linear Regression** → One independent variable.
- **Multiple Linear Regression** → More than one independent variable.

Logistic Regression

Uses the independent variables to estimate the probability of an event occurring (either 0 or 1)

Polynomial Regression

The regression line is curved rather than straight.

Problem Statement

Scenario: Predicting house price based on its size.

Goal: Find a line that best represents the relationship between size and price, so we can predict :
"If a new house is 100 m^2, what would its estimated price be?"

House Size (m^2)	Price ($)
50	150,000
70	200,000
90	250,000
110	300,000

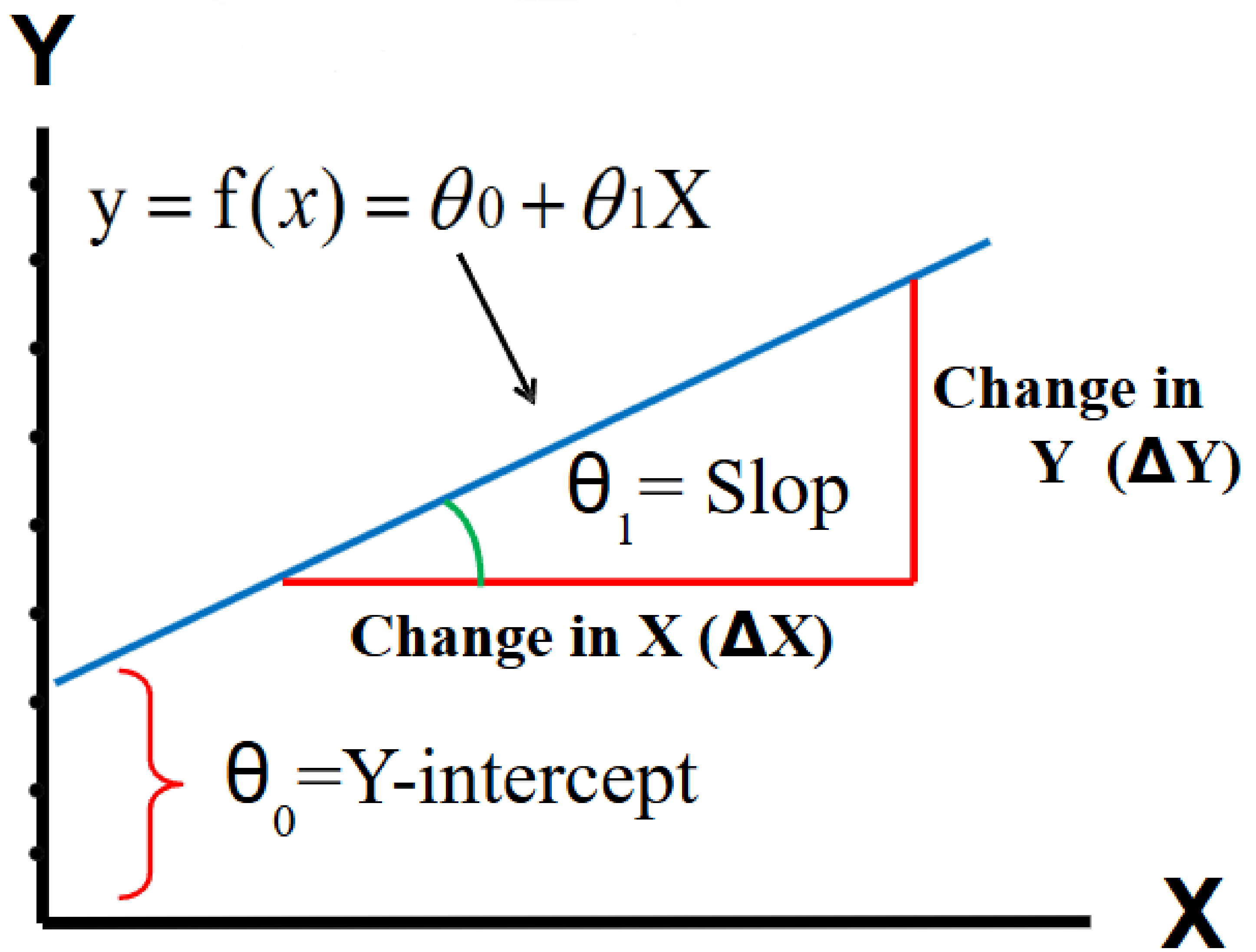
Hypothesis Function

Mathematical Form

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

Where:

- $h_\theta(x)$: the predicted value (hypothesis)
- θ_0: intercept (the value of y when $x = 0$)
- θ_1: slope (how much y changes for each unit change in x)
- x: input variable



Cost Function

The Cost Function measures how well our hypothesis Function fits the data .
It calculate the difference between the predicted value $\hat{Y}$ and the true value Y , also called **Loss Function**.

In Linear Regression**, the Mean Squared Error (MSE)** cost function is employed, which calculates the average of the squared errors between the predicted values $\hat{y}_i$ and the actual values y_i .The purpose is to determine the optimal values for the intercept θ_1 and the coefficient of the input feature θ_2 providing the best-fit line for the given data points. The linear equation expressing this relationship is $\hat{y}_i = \theta_1 + \theta_2 x_i$.

MSE function can be calculated as:

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

Where:

m: number of training examples

- $h_\theta(x^{(i)})$:predicted value for the i-th example
- $y^{(i)}$:actual value for the i-th example

- The goal is to **minimize** the cost function $J(\theta_0, \theta_1)$to find the **best-fitting line**

Cost function visualization

Consider a simple case of hypothesis by setting θ_0=0, then ***h*** becomes $\mathbf{h_\theta(x)=\theta_1 x}$

Each value of θ_1 corresponds to a different hypothesis as it is the **slope** of the line

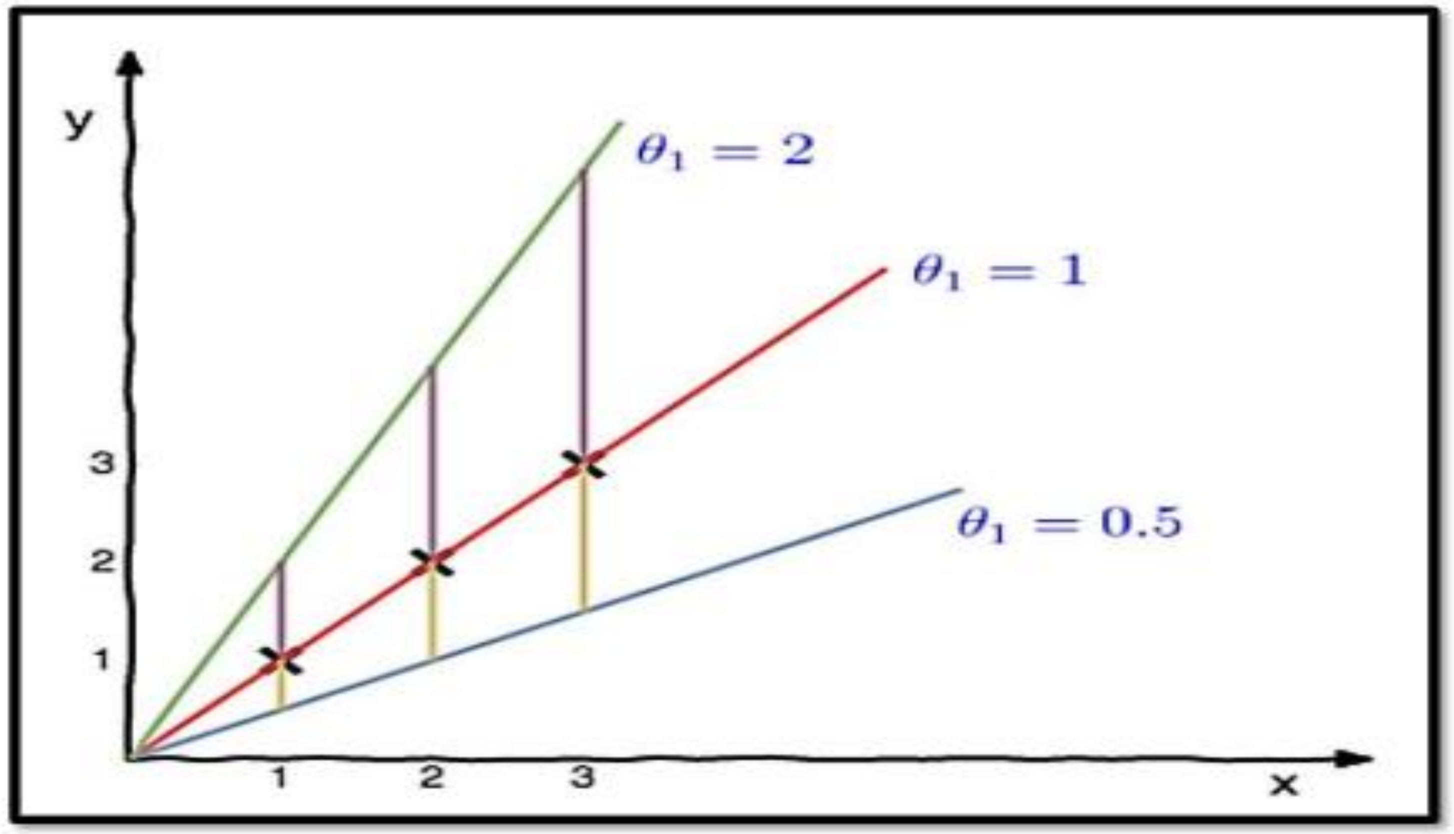
which corresponds to different lines passing through the **origin** as shown in plots below as **y-intercept** i.e. θ_0 is nulled out.

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

At θ_1=2, $J(2) = \frac{1}{2*3}(1^2 + 2^2 + 3^2) = \frac{14}{6} = 2.33$

At θ_1=1, $J(1) = \frac{1}{2*3}(0^2 + 0^2 + 0^2) = 0$

At θ_1=0.5, $J(0 = \frac{1}{2*3}(0.5^2 + 1^2 + 1.5^2) = 0.58$



Cost function visualization

What is the optimal value of θ_1 that minimizes $J(\theta_1)$?

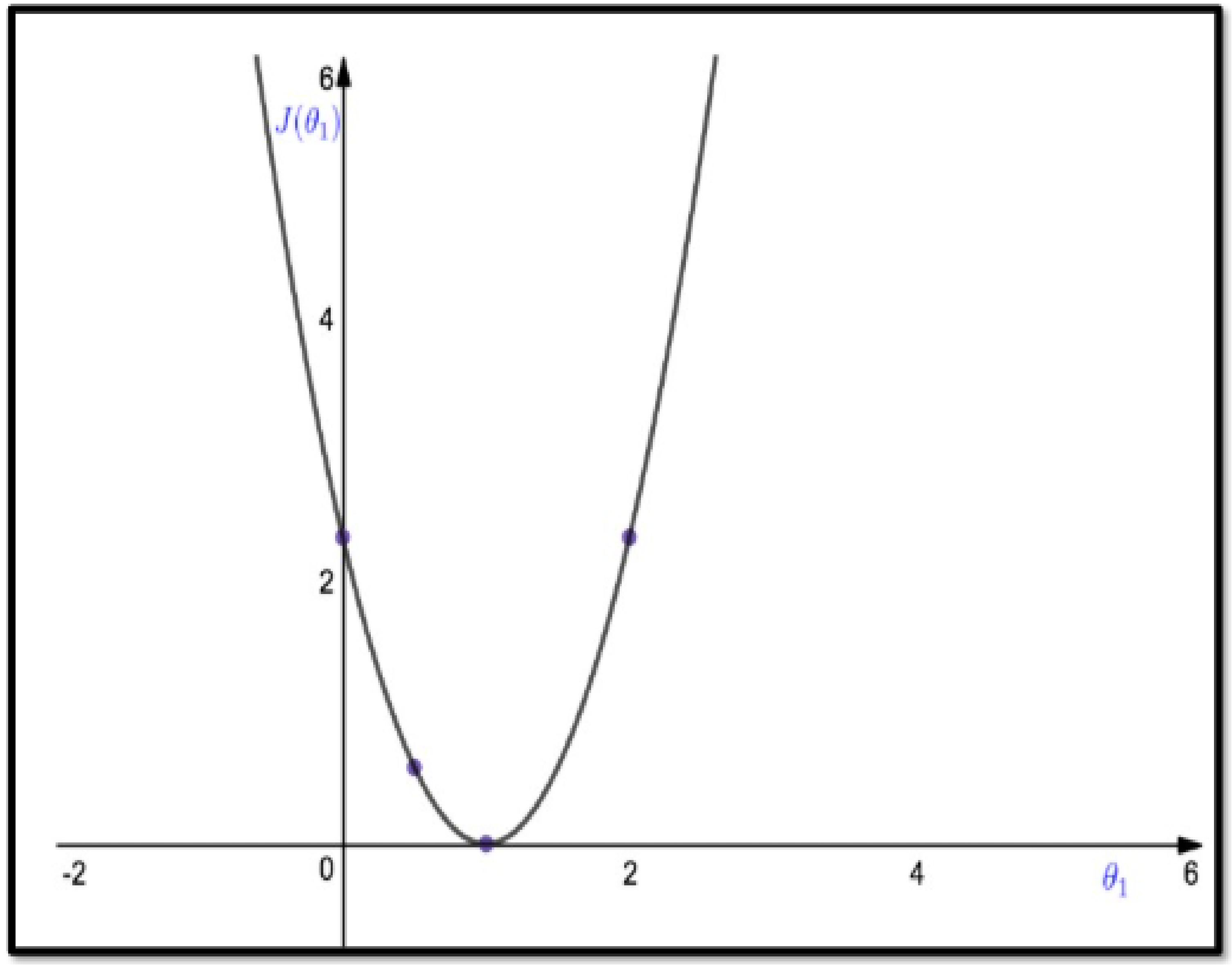
t is clear that best value for $\theta_1 = 1$ as $J(\theta_1) = 0$, which is the minimum.

How to find the best value for θ_1 ?

Plotting ?? Not practical specially in high dimensions?

The solution :

1.Analytical solution: not applicable for large datasets
2.Numerical solution: ex: Gradient descent .



Cost Function

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

Parameters: θ_0, θ_1

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Gradient Descent

Gradient Descent

- Is an Optimization algorithm used to minimize the cost function.
- It updates the parameters θ_0 , θ_1 step by step in the direction that reduce the errors.
- The process continues until the cost function reaches its minimum value.
- Iterative Solution not only in Linear Regression , it’s actually used all over the place in machine learning .

Gradient Descent

Have some function $J(\theta_0\theta_1,)$

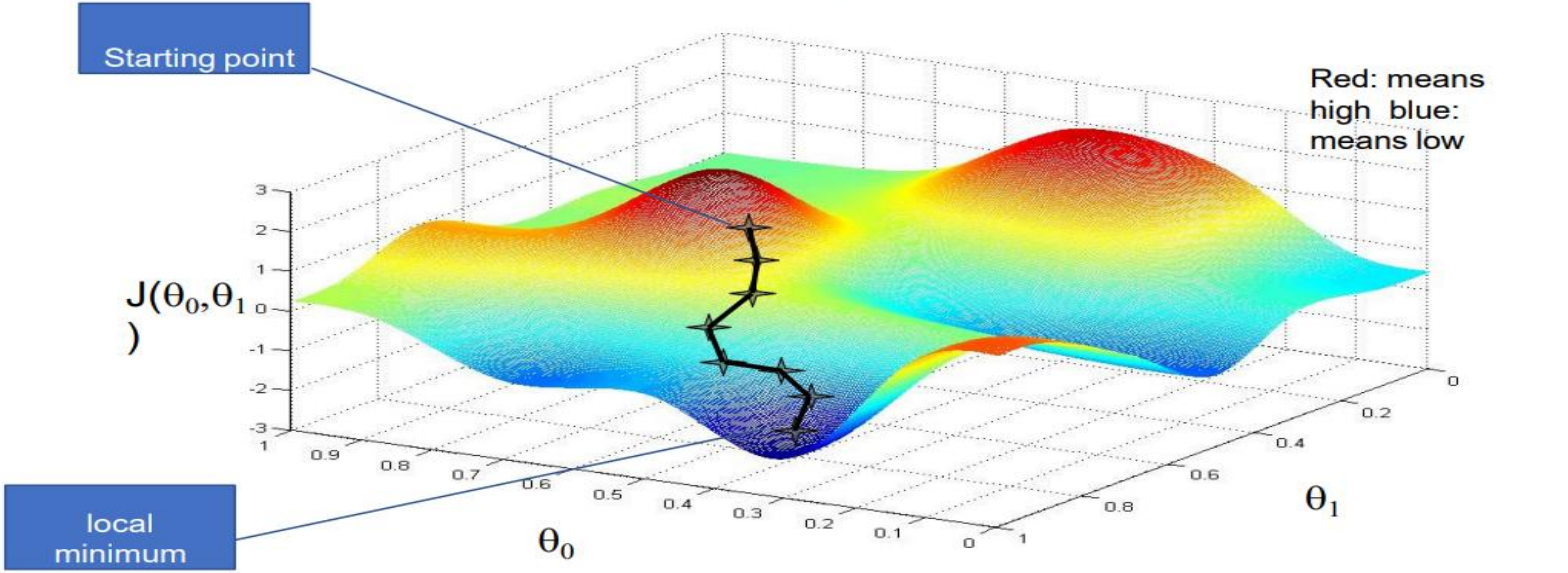
Want $\min J(\theta_0\theta_1,)$

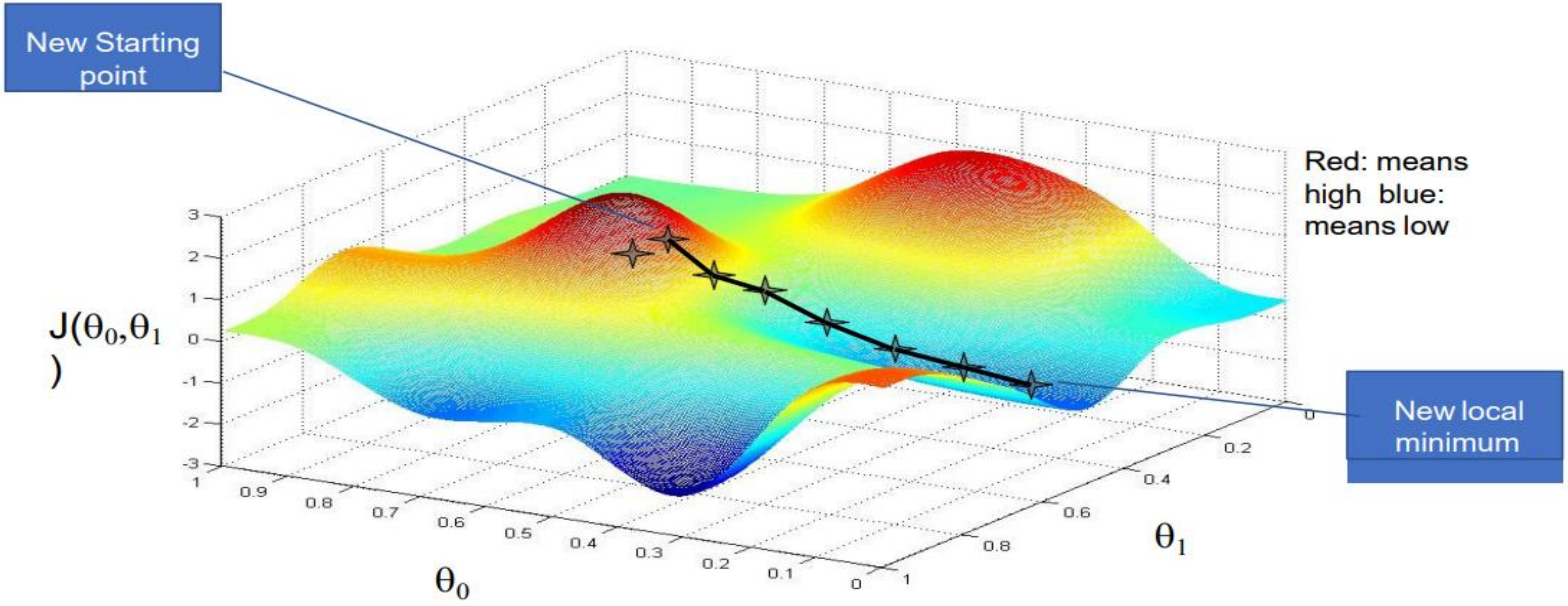
Outline :

- start with some θ_0 θ_1
- Keep changing θ_0 θ_1 $to\ reduce J(\theta_0\theta_1,)$

  until we hopefully end up at a minimum .

Imagine that this is a landscape of grassy park, and you want to go to the lowest point in the park as rapidly as possible





With different starting point

Gradient Descent Algorithm

$$\text{repeat until convergence}\{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \, \forall j \in \{0,1\}\}$$

- Where
  - := is the assignment operator
  - α is the **learning rate** which basically defines how big the steps are during the descent
  - $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the **partial derivative** term
  - j = 0, 1 represents the **feature index number**

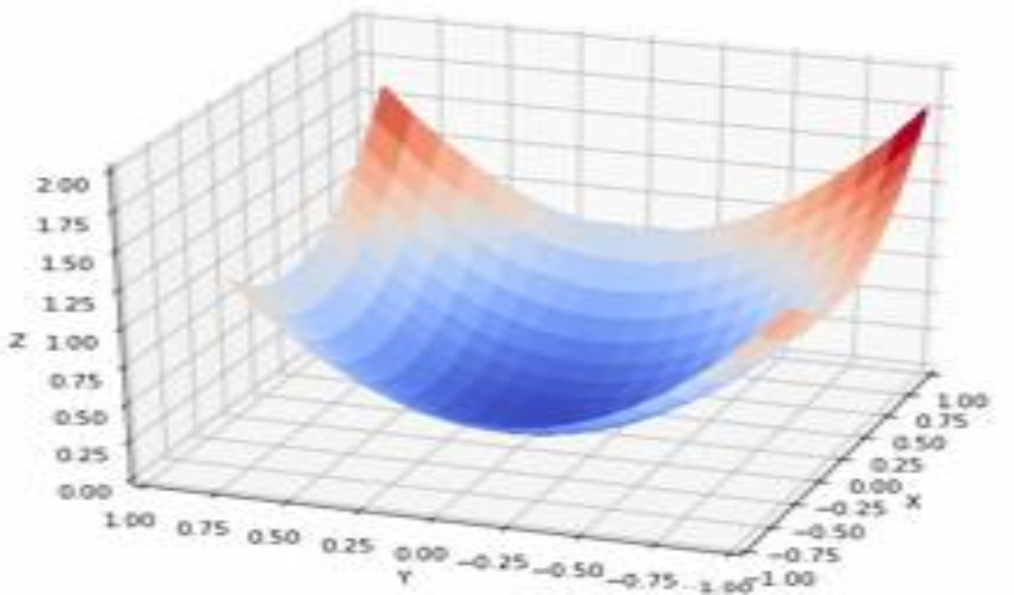
Also the parameters should be **updated simulatenously**, i.e. ,

$$temp_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

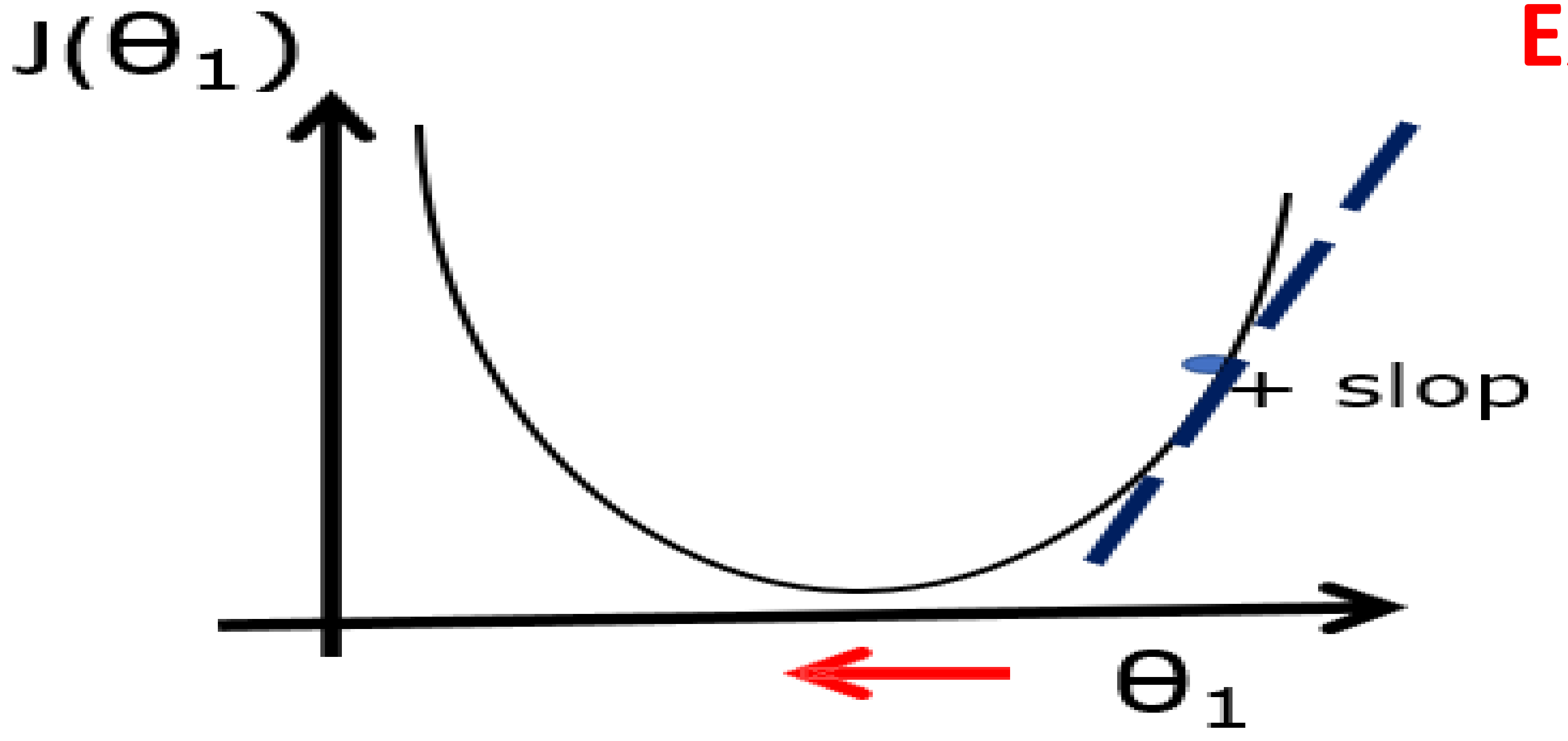
$$temp_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := temp_0$$

$$\theta_1 := temp_1$$



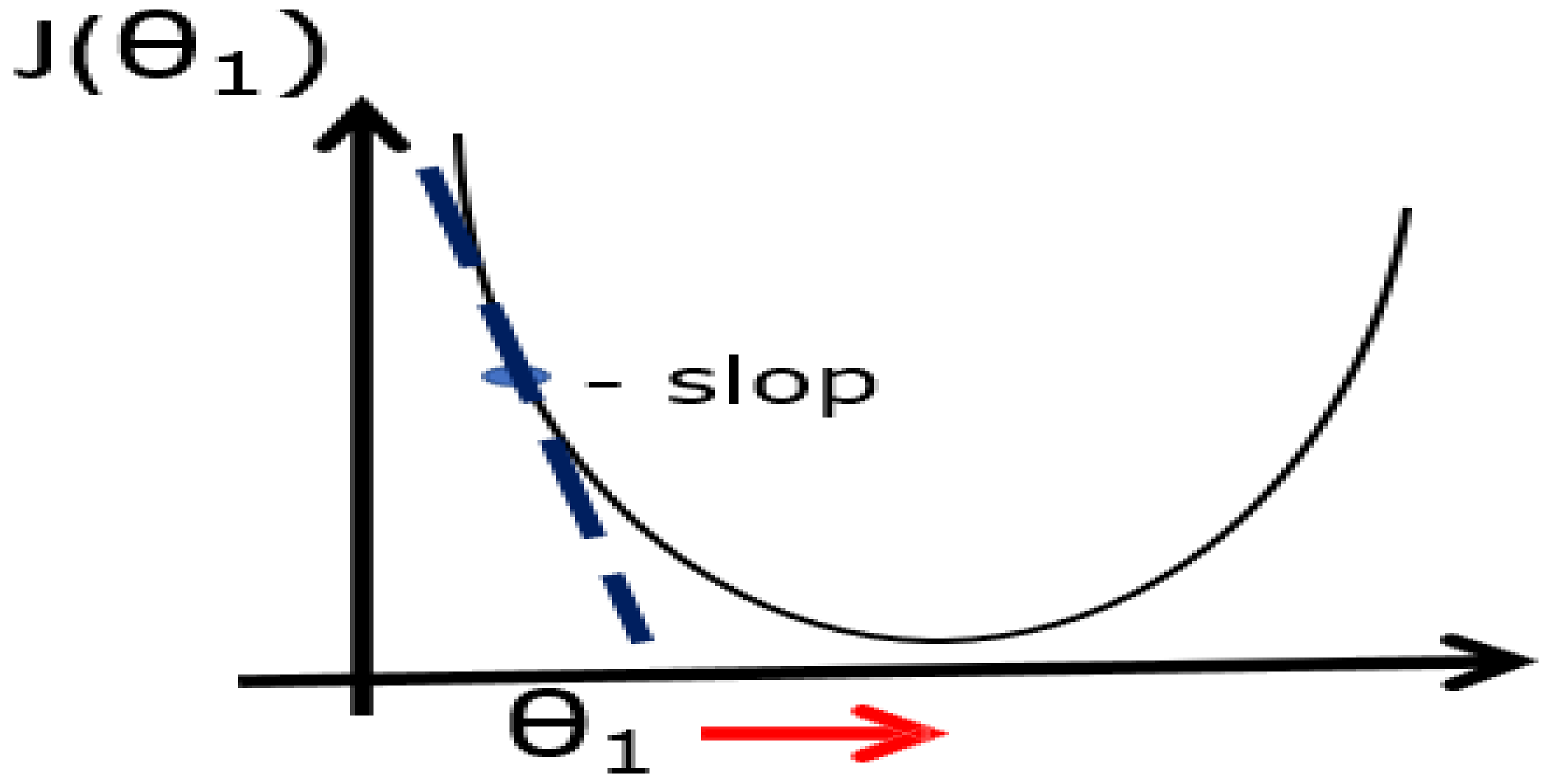
Gradient Descent Algorithm



Example

$$\boxed{\theta_1 = \theta_1 - \alpha \frac{d}{d\theta_1} j(\theta_1)}$$

$\theta_1 = \theta_1 - \alpha(+ve)$



$\theta_1 = \theta_1 - \alpha(-ve)$

Gradient Descent For a linear Regression

Gradient Descent Algorithm

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

Linear Regression Model

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

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

Gradient Descent Algorithm

$$\frac{d}{d\theta_j} j(\theta_0,\theta_1) = \frac{d}{d\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h\theta(x_i) - Y_i\right)^2$$

$$\frac{d}{d\theta_j} j(\theta_0,\theta_1) = \frac{d}{d\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 (x_i) - Y_i\right)^2$$

$$j = 0: \frac{d}{d\theta_0} j(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - Y_i\right)$$

$$j = 1: \frac{d}{d\theta_1} j(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - Y_i\right) \bullet x_i$$

Gradient Descent Algorithm

repeat until convergence {

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

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

}

update θ_0 and θ_1 simultaneously

Gradient Descent Algorithm

"Batch" Gradient descent

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

$$
\text{repeat until convergence } \{
$$

$$
\left.
\begin{aligned}
\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \\
\theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}
\end{aligned}
\right\}
$$

update θ_0 and θ_1 simultaneously

$$
\}
$$

Gradient Descent Algorithm

- **Batch Learning:**

Use all training examples to update the parameters in each gradient descent algorithm.

- **Online Learning:**

Use each individual training example to update the parameters in each gradient descent algorithm.

Evaluation

Evaluation

Model Evaluation Metrics

1. Mean Squared Error (MSE)

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

- Used to measure the average squared difference between predicted and actual values.Best when you want to penalize larger errors more heavily.
- A smaller MSE value means better model performance.

2. Root Mean Squared Error (RMSE)

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

- *Provides the error in the same unit as the target variable (Y).*
- *Easier to interpret compared to MSE.*
- *Commonly used when the scale of the prediction error is important.*

Evaluation

Model Evaluation Metrics

3. R^2 Score (Coefficient of Determination)

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

$Measures\ how\ much\ of\ the\ variance\ in\ the\ target\ variable\ is\ explained\ by\ the\ model.$

$Perfect\ fit\ \rightarrow\ R^2 = 1$

$Model\ explains\ none\ of\ the\ variance\ \rightarrow\ R^2 = 0$

$Higher\ R^2\ indicates\ better\ model\ performance.$

Applications

1. Business & Economics

Predicting sales , profits ,or market tends based on advertising spend,pricing , demand .

2. Helthcare

Predicting disease progression or patient recovery time from medical .

3. Education

Predicting student performance based on study hours , attendance , and previous scores .

Applications

4. Engineering

Modeling relationships between Variables , such as temperature vs.pressure , or stress vs. strain .

5. Real Estate

Estimating house prices from features such as , size , number of room , and location .

6. Finance

Forecasting stock prices , interest rates or credit risk using historical data .

Code & Dataset

Access the implementation and data here:
https://drive.google.com/drive/folders/1b1BaCqWzDtVxIc-iBU0BoP3nZBlS_NJQ?usp=sharing

Conclusion

- Linear Regression is one of the simplest and most important algorithm in machine learning .
- It helps to find the relationship between input and output variables and predict contiuous values .
- The model works by fitting a straight line that minimizes the cost function (error) using Gradient descent .

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

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

Thank You