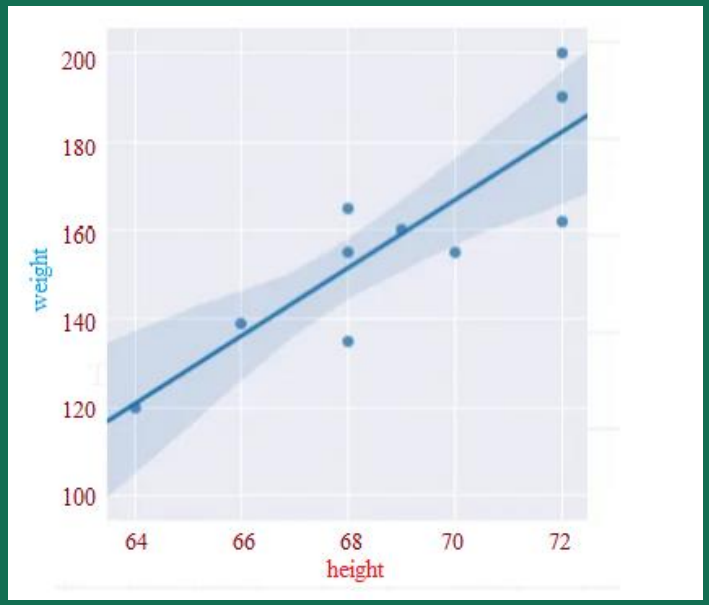
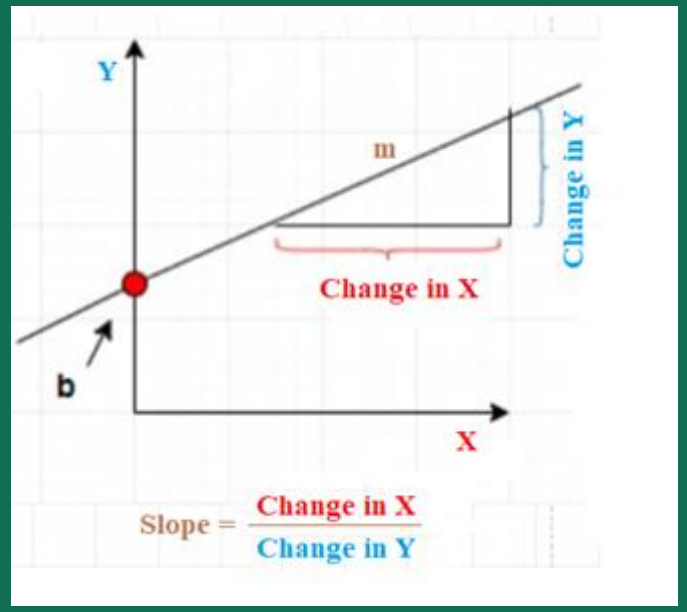
# Gradient Descent

# A Machine Learning Model

- Consider a bunch of data points that is related to the height and weight of a group of students.
- We are trying to predict some relationship between these quantities to predict the weight of some new students afterward.
- This is essentially a simple example of a supervised Machine Learning technique.



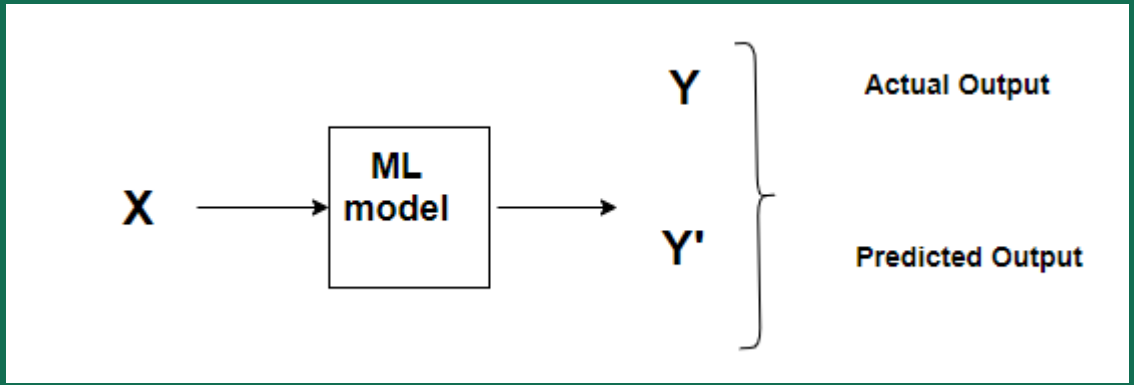




- Let us now draw an arbitrary line in space that passes through some of these data points.
- The equation of this straight line would be `Y = mX + b` where m is the slope and b is its intercept on the Y-axis

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Predictions

- Given a known set of inputs and their corresponding outputs, A machine learning model tries to make some predictions for a new set of inputs.




- The Error would be the difference between the two predictions.

$$\text{Error} = Y'(\text{Predicted}) - Y(\text{Actual})$$

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Cost Function

- A Cost Function/Loss Function evaluates the performance of our Machine Learning Algorithm.
- A Cost function basically tells us 'how good' our model is at making predictions for a given value of m and b.
- The Loss function computes the error for a single training example, while the Cost function is the average of the loss functions for all the training examples.

$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

*Why do we take the squared differences and simply not the absolute differences?*

Because the squared differences make it easier to derive a regression line. Indeed, to find that line we need to compute the first derivative of the Cost function, and it is much harder to compute the derivative of absolute values than squared values. Also, the squared differences increase the error distance, thus, making the bad predictions more pronounced than the good ones.

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

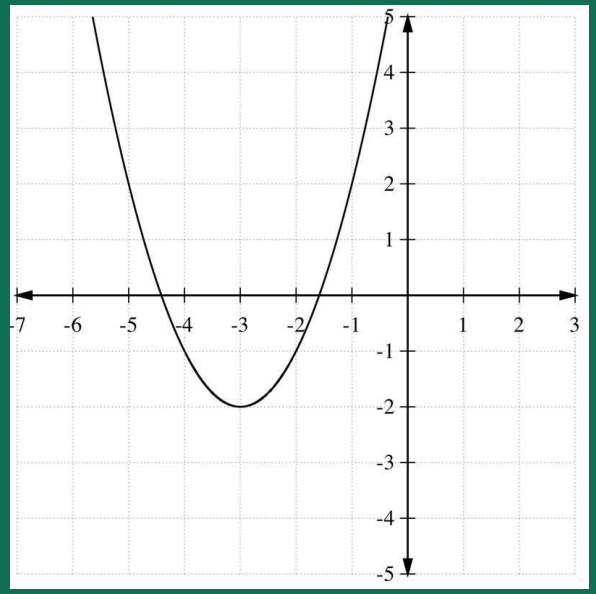
# Minimizing the Cost Function

- The goal of any Machine Learning Algorithm is to minimize the Cost Function.

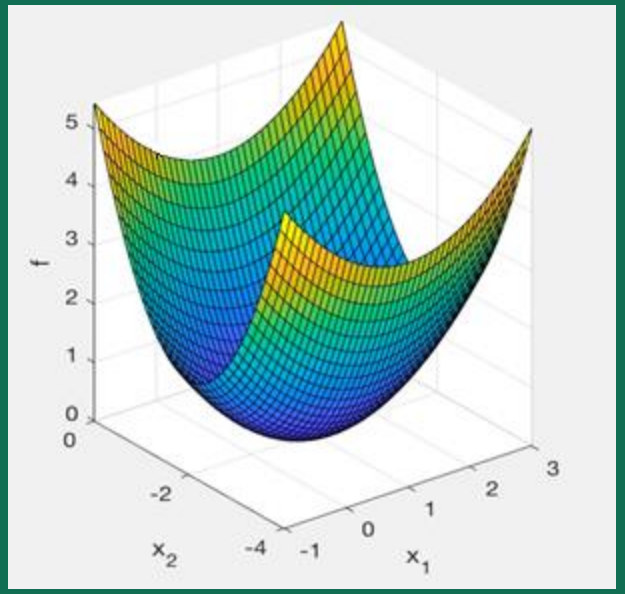
$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

**How do we minimize any function?**

- If we look carefully, our Cost function is of the form $Y = X^2$.
- In a Cartesian coordinate system, this is an equation for a parabola.
- To minimize the function above, we need to find that value of X that produces the lowest value of Y which is -3 in this example.




- It is pretty easy to locate the minima in 2D graph, but tricky in higher dimensions.
- For those cases, we need to devise an algorithm to locate the minima, and that algorithm is called Gradient Descent.




Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Gradient Descent

i. In the equation, $y = mX+b$, 'm' and 'b' are its parameters.
ii. During the training process, there will be a small change in their values. Let that small change be denoted by δ.

iii. The value of parameters will be updated as $m=m-\delta_m$ and $b=b-\delta_b$, respectively.

iv. Our aim here is to find those values of m and b in $y = mx+b$ for which the error is minimum, i.e., values that minimize the cost function.

**Parameters with small changes:**

$$m = m - \delta_m$$
$$b = b - \delta_b$$

**Given Cost Function for 'N' no of samples**

$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y_i' - Y_i)^2$$

**Cost function is denoted by J where J is a function of m and b**

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Y_i' - Y_i)^2$$

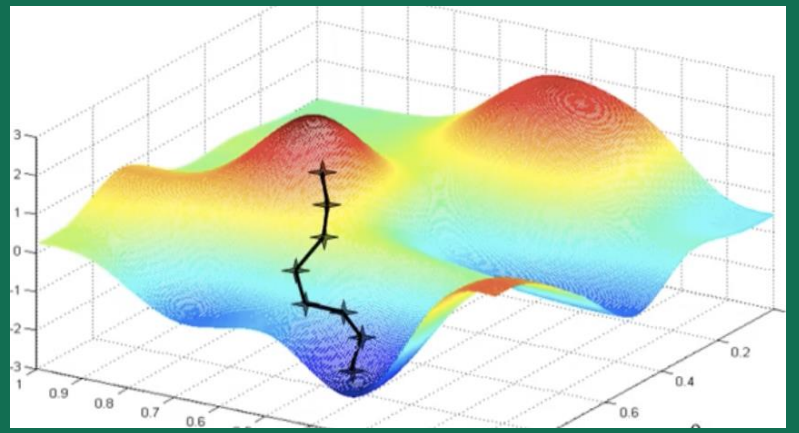
**Substituting the term Y'-Y with error for simplicity**

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

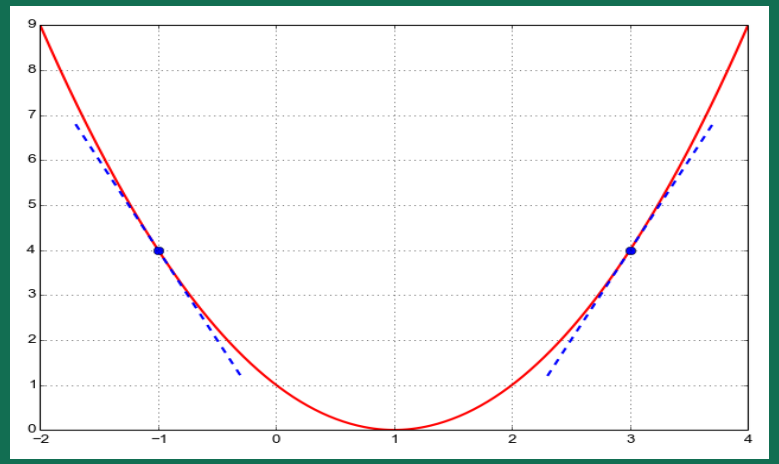
# Gradient Descent

Gradient descent is one of the most popular algorithms to perform optimization. It is an iterative optimization algorithm used to find the minimum value for a function.



There are two things that you should know to reach the minima:

1. Which way to go: you might go upward or downward
2. How big a step to take: you might take a bigger step or a little step to reach your destination.



- Gradient Descent Algorithm helps us to make these decisions efficiently and effectively with the use of derivatives.
- A derivative is calculated as the slope of the graph at a particular point.
- The slope is described by drawing a tangent line to the graph at the point.

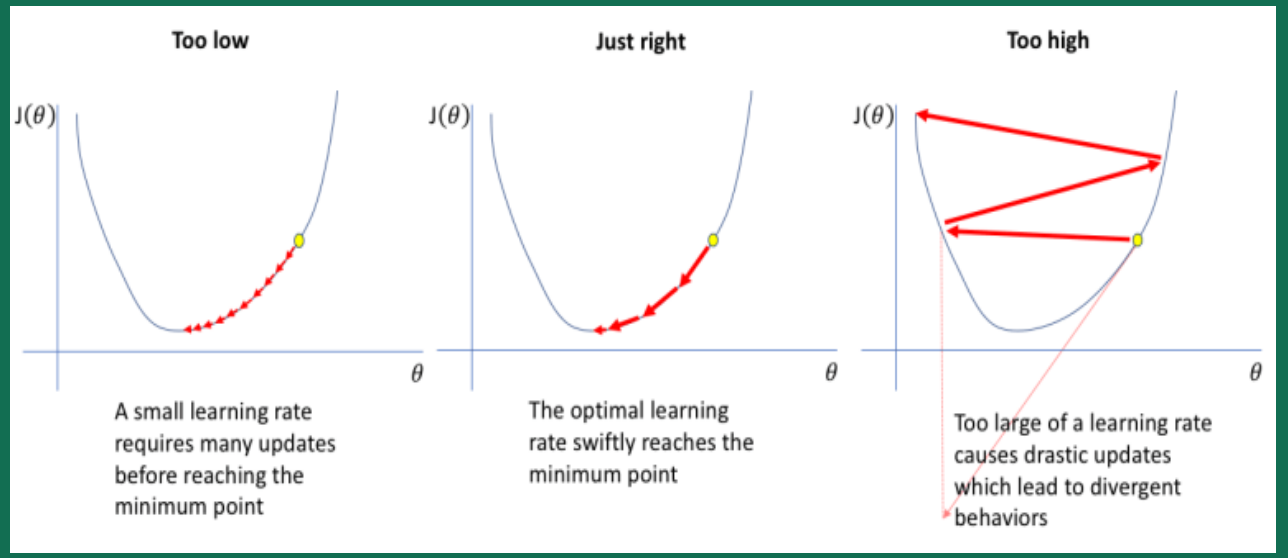
Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

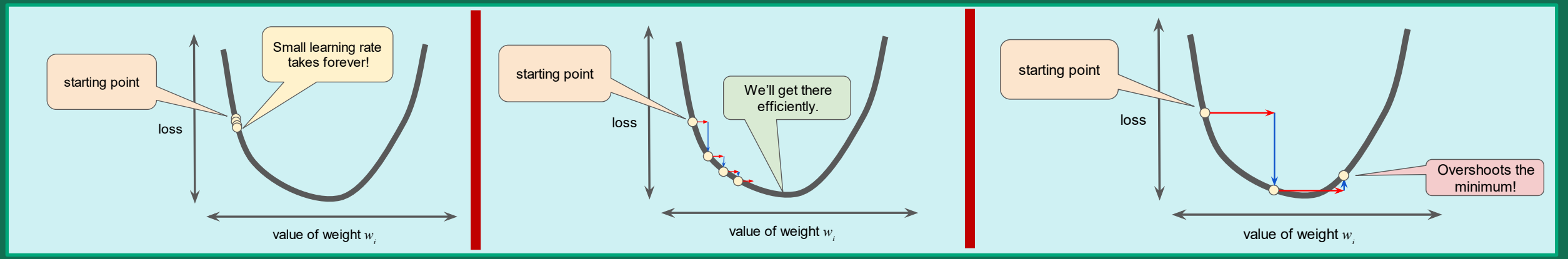
# Learning Rate

i. The gradient (vector) has both a direction and a magnitude.

ii. Gradient descent algorithms multiply the gradient by a scalar known as the learning rate (sometimes called step size) to determine the next point.

iii. For example, if the gradient magnitude is 2.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.025 (2.5*0.01) away from the previous point.




https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/fitter/graph_dd0b0587090df3c0b6d2ee48a5c42cb480db8dc97f88693f101028edd78c635c.frame

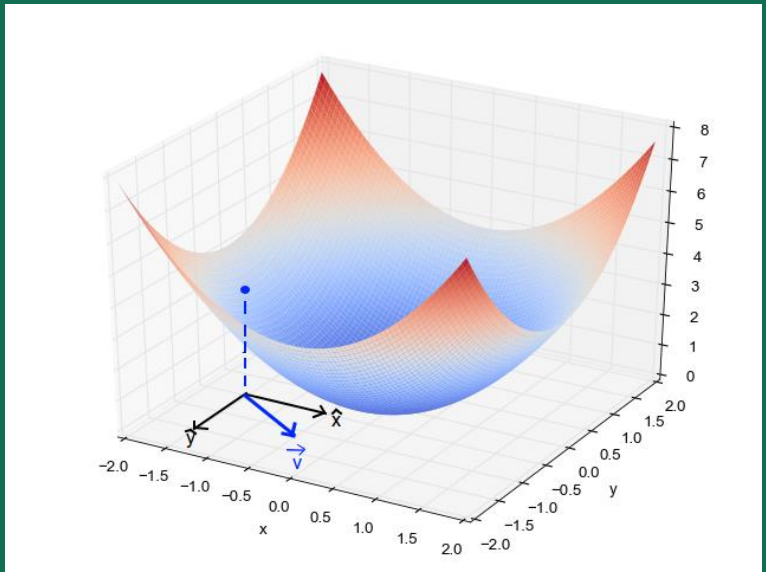
Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Derivatives

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Gradient vs Derivative

- In one variable functions: you only need the derivative to know *how a function is changing* when the variable is changing (increasing or decreasing, generally by 1 unit).
- In multivariable functions: there are multiple directions that a function can change in and therefore it is no longer sufficient to have only 1 derivative.
- So, we define the gradient to describe how the function changes in multiple dimensions. The gradient is just a vector with the function's partial derivatives for components.
- For instance, `grad(f(x,y)) = <df(x,y)/dx, df(x,y)/dy>`.
- In short, you can think of a gradient as a kind of multidimensional derivative.




Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Derivatives: Power Rule

If we have a function like

$$f(x) = x^n$$

, then

$$\frac{\partial f(x)}{\partial x} = nx^{n-1}$$

Example : Find the derivative of the function f(x) w.r.t. x where

$$f(x) = 3x^5$$

$$\frac{\partial f(x)}{\partial x} = 15x^4$$

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Derivatives: Chain Rule

if

$$y = x^2$$

and

$$x = z^2$$

then, the derivative of **y** w.r.t **z** can be calculated with **chain rule** as follows:

$$\frac{\partial y}{\partial z} = \frac{\partial y}{\partial x} . \frac{\partial x}{\partial z}$$

$$\frac{\partial y}{\partial x} = 2x$$

$$\frac{\partial x}{\partial z} = 2z$$

Hence,

$$\frac{\partial y}{\partial z} = 2x.2z$$

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Derivatives: Partial Derivatives

If there is a function of two variables, then to find the partial derivative of that function w.r.t to one variable, treat the other variable as constant.

**Partial Derivatives**

Say for instance, we have a function:

$$f(x, y) = x^4 + y^7$$

partial derivative of the function w.r.t 'x' will be :

$$\frac{\partial f}{\partial x} = 4x^3 + 0$$

treating 'y' as a constant

And partial derivative of the function w.r.t 'y' will be :

$$\frac{\partial f}{\partial y} = 0 + 7y^6$$

treating 'x' as a constant

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Calculating Gradient Descent

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

$$\frac{\partial J}{\partial m} = 2.Error.\frac{\partial}{\partial m}Error$$

$$\frac{\partial J}{\partial b} = 2 \cdot Error.\frac{\partial}{\partial b}Error$$

- For simplicity, we get rid of the summation sign.
- The summation part is important, especially with the concept of Stochastic Gradient Descent (SGD ) vs. Batch Gradient Descent (BGD).
- During the BGD, we look at the error of all the training examples at once, while in the SGD, we look at each error at a time.
- However, to keep things simple, we will assume that we are looking at each error one at a time.

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Calculating Gradient Descent

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

$$\frac{\partial J}{\partial m} = 2.Error.\frac{\partial}{\partial m}Error$$

$$\frac{\partial J}{\partial b} = 2 \cdot Error.\frac{\partial}{\partial b}Error$$

$$\frac{\partial}{\partial m}\text{Error} = \frac{\partial}{\partial m}(Y' - Y)$$

$$\frac{\partial}{\partial m}\text{Error} = \frac{\partial}{\partial m}(mX + b - Y)$$

constants

$$\frac{\partial}{\partial m}\text{Error} = X$$

$$\frac{\partial}{\partial b}\text{Error} = \frac{\partial}{\partial b}(Y' - Y)$$

$$\frac{\partial}{\partial b}\text{Error} = \frac{\partial}{\partial b}(mX + b - Y)$$

constants

$$\frac{\partial}{\partial b}\text{Error} = 1$$

$$\frac{\partial J}{\partial m} = 2.\text{Error} * X * \text{Learning Rate}$$

Determines the direction to minimize the Error

Determines how large a step to take

This 2 in this equation isn't that significant since it just says that we have a learning rate twice as big or half as big.

$$\frac{\partial J}{\partial m} = \text{Error} * X * \text{Learning Rate}$$

$$\frac{\partial J}{\partial b} = \text{Error} * \text{Learning Rate}$$

$$Since\ m = m - \delta_m$$

$$Since\ b = b - \delta_b$$

$$m^1 = m^0 - \text{Error} * X * \text{Learning Rate}$$

$$b^1 = b^0 - \text{Error} * \text{Learning Rate}$$

$m^1$, $b^1$ = next position parameters; $m^0$, $b^0$ = current position parameters

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Same thing in Different Form

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

Parameters: $\theta_0, \theta_1$

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

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

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)}\right)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)}\right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)}\right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)}\right) x_j^{(i)}
\end{aligned}
$$

repeat until convergence {

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

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

repeat until convergence {

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

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

}

Source: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

# Thank You.