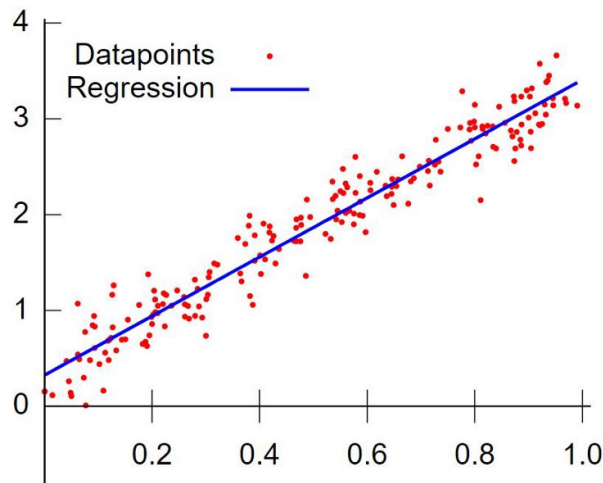


Linear Regression

Linear regression is a linear model that establishes the relationship between a dependent variable $y(\text{Target})$ and one or more independent variables denoted $X(\text{Inputs})$.



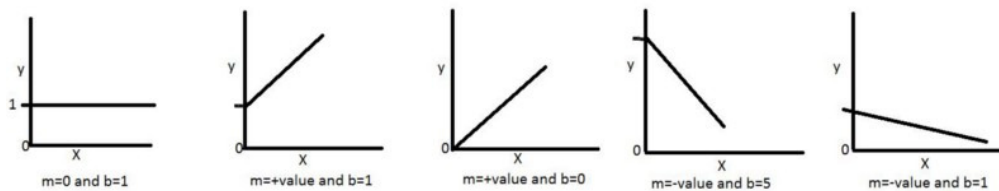
Goal is to find that blue straight line (which is best fit) to the data.

Our Training Data consists of X and y values so we can plot them on the graph, that's damn easy. Now how to find that blue line?

First let's talk about how to draw a linear line in the graph,

In math we have an equation which is called linear equation

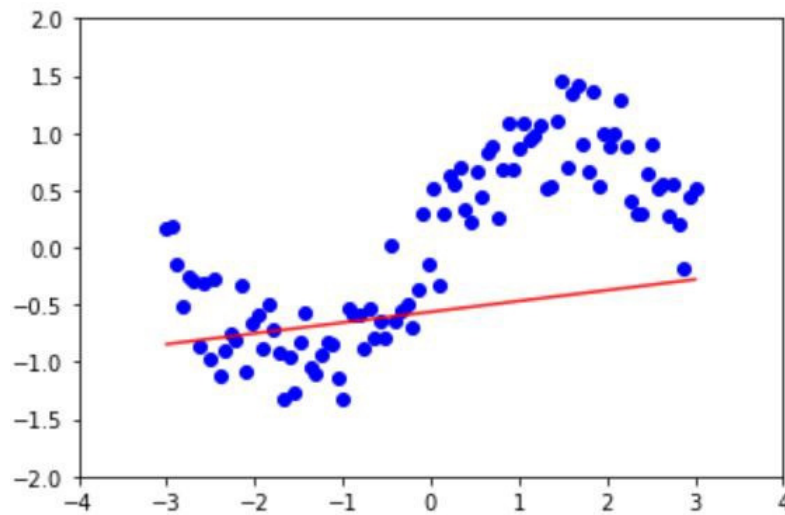
$$y = mX + b \quad \{ m \rightarrow \text{slope}, b \rightarrow \text{Y-intercept} \}$$



So we can draw the line if we take any values for m and b

How do we get the m and b values and how do we know exact m and b values for the best fit line??

Let's take a simple data set (sine wave form -3 to 3) and First time we take random values of m and b values and we draw a line something like this.



Random line for m and b

How we drew the above line?

We take the first X value(x_1) from our data set and calculate y value (y_1)

$y_1 = m \cdot x_1 + b$ {m,b->random values lets say 0.5,1

$x_1 \rightarrow$ lets say -3 (first value from our data-set)

$y_1 = (0.5 \cdot -3) + 1$

$y_1 = -0.5$

by applying all x values for m and b values we get our first line.

Above picture has its own random variables (I hope you understand the concept)

That line is *not* fitting well to the data so we need to change m and b values to get the best fit line.

How do we change m and b values for the best fit line??

1. Either we can use an awesome algorithm called **Gradient Descent** (Which I will cover in next story with also the math used in there.)
2. We can borrow direct formulas from statistics (they call this **Least Square Method**).

$$m = \frac{\sum_{i=1}^n (x_i - \bar{X})(y_i - \bar{Y})}{\sum_{i=1}^n (x_i - \bar{X})^2} \quad b = \bar{Y} - m\bar{X}$$

\bar{X} is mean of X values, \bar{Y} mean of y values

3.

Regression model in matrix form

The linear model with several explanatory variables is given by the equation

$$y_i = \beta_1 + \beta_2 x_{2i} + \beta_3 x_{3i} + \cdots + \beta_k x_{ki} + \varepsilon_i \quad (i = 1, \dots, n). \quad (3.1)$$

From now on we follow the convention that the constant term is denoted by β_1 rather than α . The first explanatory variable x_1 is defined by $x_{1i} = 1$ for every $i = 1, \dots, n$, and for simplicity of notation we write β_1 instead of $\beta_1 x_{1i}$. For purposes of analysis it is convenient to express the model (3.1) in *matrix form*. Let

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{21} & \cdots & x_{k1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{2n} & \cdots & x_{kn} \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}. \quad (3.2)$$

Note that in the $n \times k$ matrix $X = (x_{ji})$ the first index j ($j = 1, \dots, k$) refers to the variable number (in columns) and the second index i ($i = 1, \dots, n$) refers to the observation number (in rows). The notation in (3.2) is common in econometrics (whereas in books on linear algebra the indices i and j are often reversed). In our notation, we can rewrite (3.1) as

$$y = X\beta + \varepsilon. \quad (3.3)$$

Here β is a $k \times 1$ vector of unknown parameters and ε is an $n \times 1$ vector of unobserved disturbances.

Residuals and the least squares criterion

If b is a $k \times 1$ vector of estimates of β , then the estimated model may be written as

$$y = Xb + e.$$

Here e denotes the $n \times 1$ vector of residuals, which can be computed from the data and the vector of estimates b by means of

We denote transposition of matrices by primes ($'$)—for instance, the transpose of the residual vector e is the $1 \times n$ matrix $e' = (e_1, \dots, e_n)$. To determine the least squares estimator, we write the sum of squares of the residuals (a function of b) as

$$\begin{aligned} S(b) &= \sum e_i^2 = e'e = (y - Xb)'(y - Xb) \\ &= y'y - y'Xb - b'X'y + b'X'Xb. \end{aligned}$$

Derivation of least squares estimator

The minimum of $S(b)$ is obtained by setting the derivatives of $S(b)$ equal to zero. Note that the function $S(b)$ has scalar values, whereas b is a column vector with k components. So we have k first order derivatives and we will follow the convention to arrange them in a column vector. The second and third terms of the last expression in (3.6) are equal (a 1×1 matrix is always symmetric) and may be replaced by $-2b'X'y$. This is a linear expression in the elements of b and so the vector of derivatives equals $-2X'y$. The last term of (3.6) is a quadratic form in the elements of b . The vector of first order derivatives of this term $b'X'Xb$ can be written as $2X'Xb$. The proof of this result is left as an exercise (see Exercise 3.1). To get the idea we consider the case $k = 2$ and we denote the elements of $X'X$ by c_{ij} , $i, j = 1, 2$, with $c_{12} = c_{21}$. Then $b'X'Xb = c_{11}b_1^2 + c_{22}b_2^2 + 2c_{12}b_1b_2$. The derivative with respect to b_1 is $2c_{11}b_1 + 2c_{12}b_2$ and the derivative with respect to b_2 is $2c_{12}b_1 + 2c_{22}b_2$. When we arrange these two partial derivatives in a 2×1 vector, this can be written as $2X'Xb$. See Appendix A (especially Examples A.10 and A.11 in Section A.7) for further computational details and illustrations.

The least squares estimator

Combining the above results, we obtain

$$\frac{\partial S}{\partial b} = -2X'y + 2X'Xb.$$

The least squares estimator is obtained by minimizing $S(b)$. Therefore we set these derivatives equal to zero, which gives the *normal equations*

$$X'Xb = X'y.$$

Solving this for b , we obtain

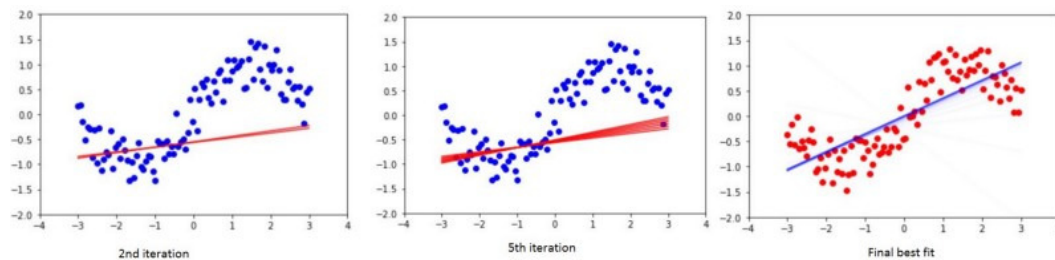
$$b = (X'X)^{-1}X'y \quad (3.9)$$

provided that the inverse of $X'X$ exists, which means that the matrix X should have rank k . As X is an $n \times k$ matrix, this requires in particular that $n \geq k$ —that is, the number of parameters is smaller than or equal to the number of observations. In practice we will almost always require that k is considerably smaller than n .

In this least square method direct calculation of $(XX)^{-1}$ is the most complex part because if data is input very huge then this matrix becomes very big and finding inverse of this matrix is very complex and time consuming.

So if we have small input data then we can use direct formula above to calculate the value of weights but if input data is huge then we will go for the Gradient descent method which works pretty fast on big input data.

Right now let's black box, we assume that we are getting the m and b values, Every time when the m and b values change we may get a different line and finally we get the best fit line



So *What's next???*

Now we can Predict new data, remember so we give new X values we get the predicted y values how does it work?

Same as above $y = mX + b$, we now know the final m and b values.

This is called **simple linear regression** as we have only one independent X value. Lets say we want predict housing price based the size of house

X= Size (in sqft's) y= Price (in dollars)

X	y
1000	40
2000	70
500	25
.....	

What if we have more independent values of X?

Let's say we want predict housing price not only by the size of house but also by no of bedrooms

x1= Size (in sqft's), x2=N_rooms and y= Price (in dollar's)

x1	x2	y
1000	2	50
2000	4	90
500	1	35
.....		

The process same as above but the equation changes a bit

Note: Lets alias b and m as θ_0 and θ_1 (theta 0 and theta 1) respectively.

$y = \theta_0 + \theta_1 * X \rightarrow b + mX \rightarrow$ Simple LR \rightarrow Single variable LR

$y = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + \dots + \theta_n * x_n \rightarrow$ Multiple LR \rightarrow Multi variable LR

Now we can predict as many things as we wish.

Maths behind Gradient Descent

In Linear Regression we need to update m and b values, we call them **weights** in machine learning. Lets alias b and m as θ_0 and θ_1 (theta 0 and theta 1) respectively.

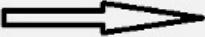
First time we take random values for θ_0 and θ_1 , and we calculate y

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

In machine learning we say hypothesis so $h(x) = \theta_0 + \theta_1 * x$

$h(x)=y$ but this y is not actual value in our data-set, this is predicted y from our hypothesis.

For example lets say our data-set is something like below and we take random values which are **1** and **0.5** for θ_0 and θ_1 respectively.

X	y		$h(x) = \theta_0 + \theta_1 * x$	predicted y
10	5		$1 + 0.5 * 10$	6
12	6.6			
3	1			
...	...	{ Actual y value is 5 and predicted y value is 6 }		

From this we calculate the error which is

$$\text{error} = (h(x) - y)^2 \rightarrow (\text{Predicted} - \text{Actual})^2$$

$$\text{error} = (6 - 5)^2 = 1$$

² is to get rid of negative values (what if Actual $y=6$ and $P_y=5$)

we just calculated the error for one data point in our data-set , we need to repeat this for all data points in our data set and sum up the all errors to one error which is called **Cost Function** ' $J(\theta)$ ' in machine learning.

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

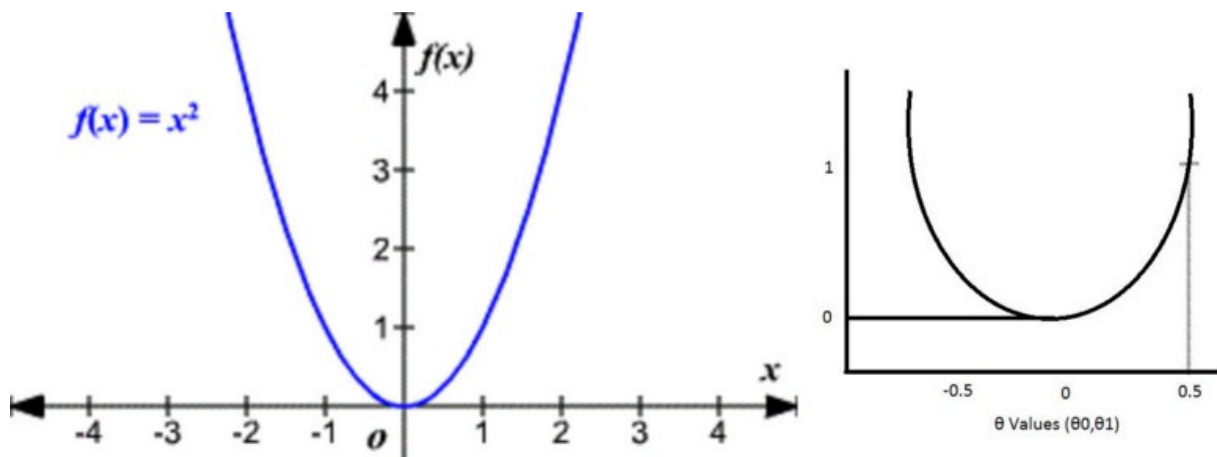
Cost Function.

- m The number of training examples
- $x^{(i)}$ The input vector for the i^{th} training example
- $y^{(i)}$ The class label for the i^{th} training example
- θ The chosen parameter values or “weights” ($\theta_0, \theta_1, \theta_2$)
- $h_{\theta}(x^{(i)})$ The algorithm’s prediction for the i^{th} training example using the parameters

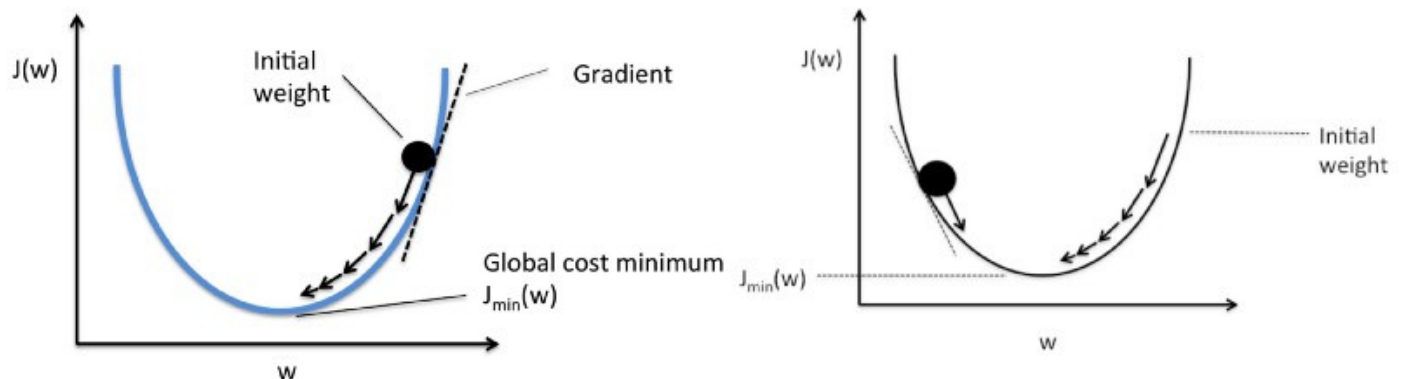
Our goal is to minimize the cost function (error) **we want our error close to zero Period.**

we have the error **1** for first data-point so lets treat that as whole error and reduce to zero for sake of understanding.

for $(h(x)-y)^2$ function we get always positive values and graph will look like this(Left) and lets plot the error graph.



Here is the gradient descent work comes into the picture.



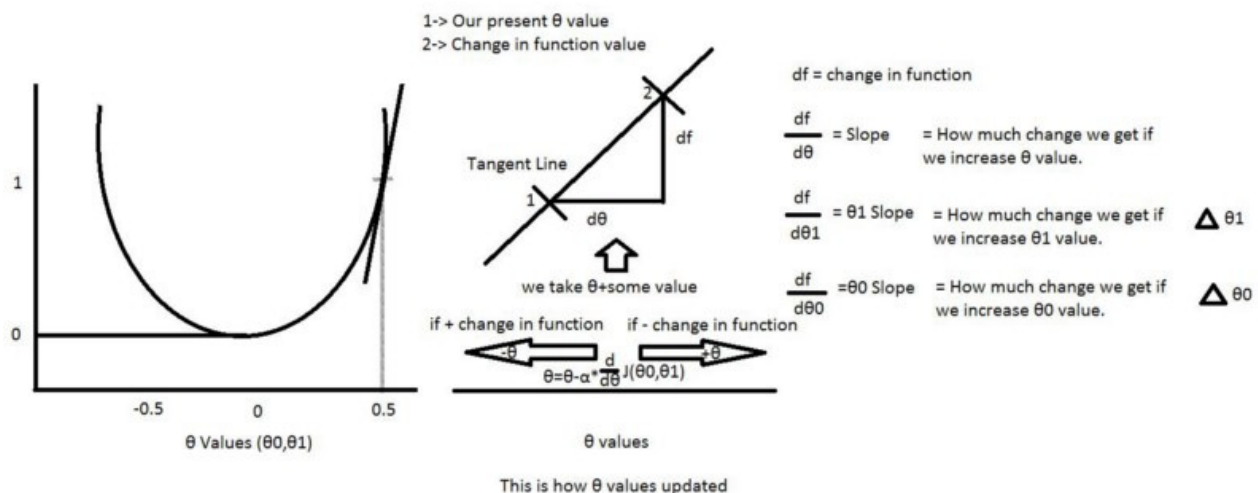
+ θ values (Left), - θ values(Right)

By taking the little steps down to reach the minimum value (bottom of the curve) and changing the θ values in the process.

How does it know how much value it should go down???

The answer is in Math.

1. It draws the line(Tangent) from the point.
2. It finds the slope of that line.
3. It identifies how much change is required by taking the partial derivative of the function with respect to θ
4. The change value will be multiplied with a variable called **alpha**(learning rate) we provide the value for alpha usually 0.01
5. It subtracts this change value from the earlier θ value to get new θ value .



From above picture we can define our θ_0 and θ_1 .

And alpha here is a learning rate usually we give 0.01 but it depends, it tells how big the step-size is towards reaching the minimum value.

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

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

Derivatives:

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

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

Repeat until convergence

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

θ_0 and θ_1 values(Left), more than two θ 's (Right)

Again we know our $J(\theta_0, \theta_1)$ so if we apply this to above equations for θ_0 and θ_1 , we get our new θ_0 and θ_1 values.

How to calculate the derivatives???

For example $f(x) = x^2 \rightarrow df/dx = 2x$ How ???

$$x^n = n * x^{n-1}$$

How to calculate the partial derivatives???

its same as calculating derivatives but here we calculate the derivative with respect to that value , others are constants (so $d/dx(\text{constant})=0$)

$$\frac{d}{d\theta_1} J(\theta_1, \theta_2) = \frac{d}{d\theta_1} \theta_1^2 + \cancel{\frac{d}{d\theta_1} \theta_2^2} = 2\theta_1$$

$$\frac{d}{d\theta_2} J(\theta_1, \theta_2) = \cancel{\frac{d}{d\theta_2} \theta_1^2} + \frac{d}{d\theta_2} \theta_2^2 = 2\theta_2$$

The same thing we can apply for calculating partial derivative with respect to **θ_0** and **θ_1** .

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

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

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

Product Rule

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

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

How come that box drawn disappeared in the next step above? Check below

For calculating partial derivative with respect to θ_1 is also same as above except one little part is added

$$\frac{d}{d\theta_0} (h_{\theta}(x^{(i)}) - y^{(i)}) = \frac{d}{d\theta_0} (\overset{1}{\theta_0} + \overset{0}{\theta_1} x^{(i)} - \overset{0}{y^{(i)}}) = 1$$

$\uparrow \quad \quad \uparrow \quad \quad \uparrow$

$$\frac{d}{d\theta_1} (h_{\theta}(x^{(i)}) - y^{(i)}) = \frac{d}{d\theta_1} (\overset{0}{\theta_0} + \theta_1 x^{(i)} - \overset{0}{y^{(i)}}) = x^{(i)}$$

$\uparrow \quad \quad \quad \uparrow$

θ_0 box disappeared because value is 1 (Top)

So Final picture is

Derivatives:

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

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

These are final **θ_0** and **θ_1** values

In the same way value for all the weights are calculated using gradient descent for all the input records and then average is taken. This average is subtracted from previous weight wrt learning rate α .

This cycle goes on until error is minimized and after that we get the final resultant line or hyper plane that we use for prediction.