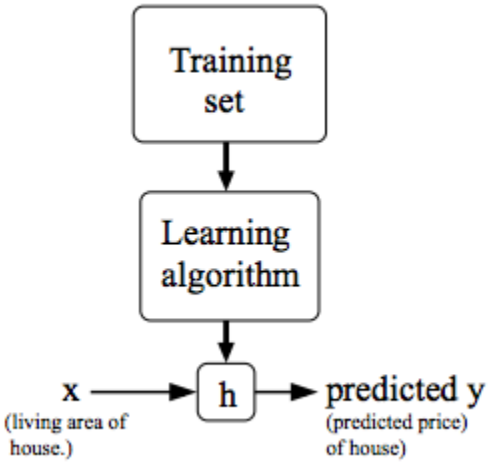
Linear Regression using Single Variable / Univariant Linear Regression :

In this we will have data set (training set) to build a model that find the relation between our input variable, X and output Variable y, so that our model predicts the value of y for unseen value of X.

Motivation Example : To predict housing price based on the size in sq. Ft.

Size in sq. Ft (X)	Price($) in 1000's (y)
2104	**460**
1416	**232**
1534	**315**
852	**178**
…….	**…….**
…….	**…….**

Model Representation:



Notations :

X → training data set

Y → output

m → size of our training data set.

$(x^{(i)}, y^{(i)})$ → i^{th} training example.

h → hypothesis (h : X → y).

Here hypothesis is the function that show how X and y values are related to each other.

Representation of h: $\mathbf{h_{theta}(x) = \theta_0 + \theta_1 * X.}$

Cost Function (J) :

In the above equation of h, the θ_0, θ_1 are parameters(learning parameters), now the problem is how to choose these parameters.

Idea : Our goal is to have values of the θ_0, θ_1 such that our $\mathbf{h_{theta}(x)}$ value should be close to the y value (we can check that while we are training).

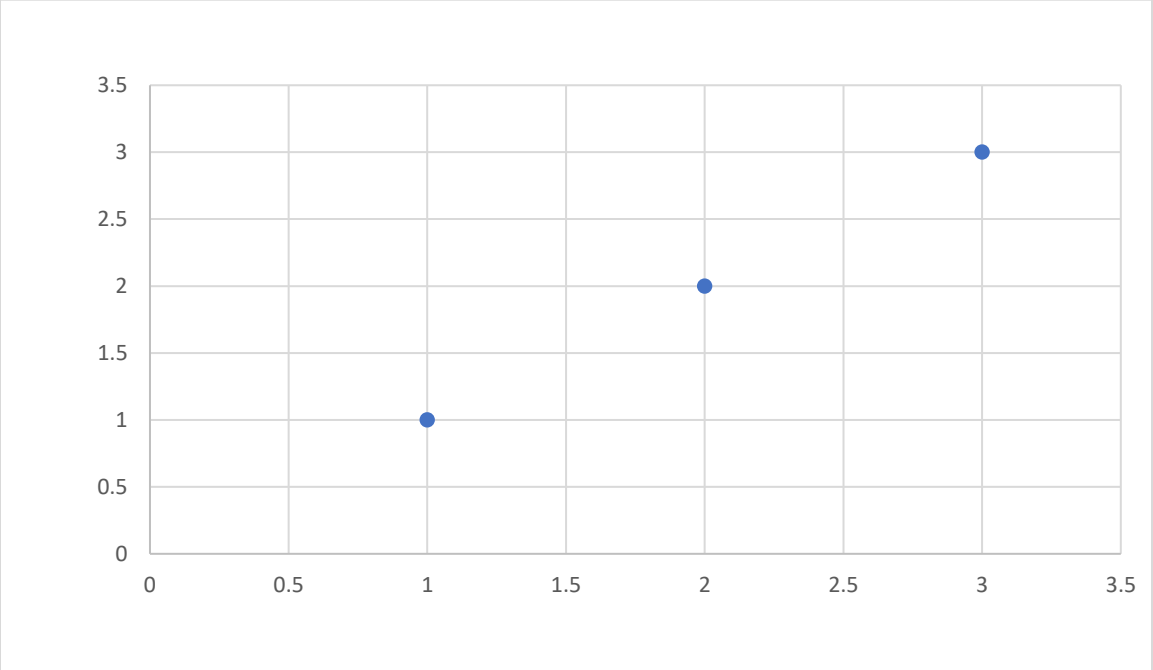
So, the difference → $\mathbf{h_{theta}(x) - y}$ should be minimum, to achieve this we can use Error functions like Mean Square Error, Root Mean Square error, ……,etc.

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

Intuition:

For better understanding let us consider $\theta_0 = 0$, so our hypothesis would be $\mathbf{h_{theta}(x) = \theta_1 * X.}$

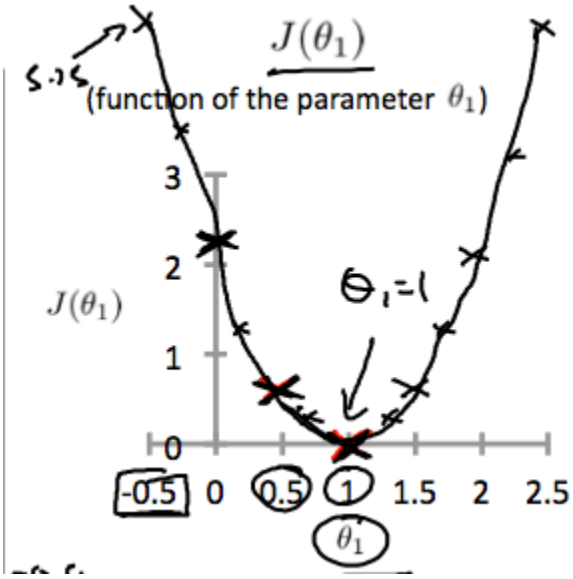
Consider the following data set.



Now calculate $\mathbf{h_{theta}(x)}$ for different values of $\boldsymbol{\theta_1}$.

- For $\boldsymbol{\theta_1} = 0$, $h_{theta}(x) = 0$ and cost function, $J(\theta_1) = 14/6$.
- For $\boldsymbol{\theta_1} = 1$, $h_{theta}(x) = 0$ and cost function, $J(\theta_1) = 0$
- For $\boldsymbol{\theta_1} = 0$, $h_{theta}(x) = 0$ and cost function, $J(\theta_1) = 0.68$.

Now if we plot a graph between $\boldsymbol{\theta_1}$ and $J(\theta_1)$ then we can get a graph look like the following.



From the above graph, we can notice that the cost function is minimum at $\theta_1 = 1$.

In the similar way we can even include θ_0 in the minimization, but for visualization we need to use a 3-D plotting, or we can use contour plots.

Reference : https://coursera.org/share/ca1bff16540cbeee4e7eb5fffd80d73e

Instead of plotting the cost function over the parameters and manually figuring out what values of parameter will yield minimum cost function, but if we have multiple parameters the manual process will be difficult, so we need an efficient algorithm to automatically figuring out the right parameter values.

Gradient descent : It is Optimization algorithm used in machine learning to tweak the parameters iteratively to minimize the given cost function (differentiable function) to its local minimum.

Steps :

1. Start with some random values parameters.
2. Keep changing the parameters until we end up at a minimum value of cost function.

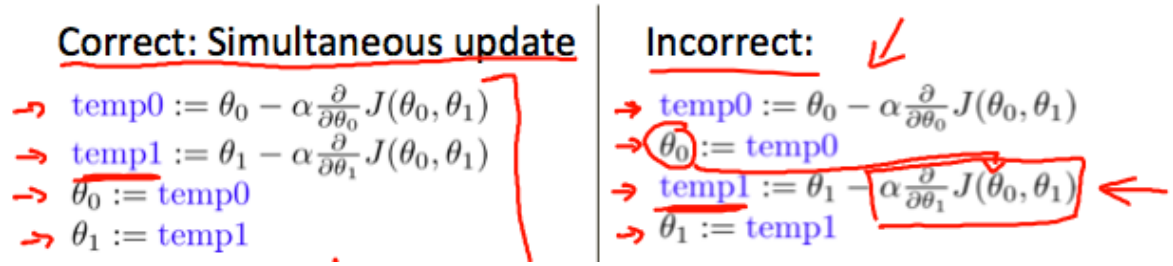
Algorithm :

Repeat until converges {

$$\Theta \mathrm{j} \;=\; \Theta \mathrm{j} \;-\; \alpha \frac{\partial \mathrm{J}}{\partial \theta}\, J(\theta_0, \theta_1, \ldots\ldots, \theta_n)$$

}

We must simultaneously update the parameter values. Suppose we have 2 parameters θ_0, θ_1 then we must update the parameters as follows :



In the incorrect one the updated theta_0 will be used in updating the theta_1 which is an error, so we follow simultaneous update in gradient descent algorithm.

Gradient Descent for Linear Regression :

Repeat until converges {

$$\theta_0 \;=\; \theta_0 - \alpha \frac{\partial}{\partial \theta 0}\, J(\theta_0, \theta_1)$$

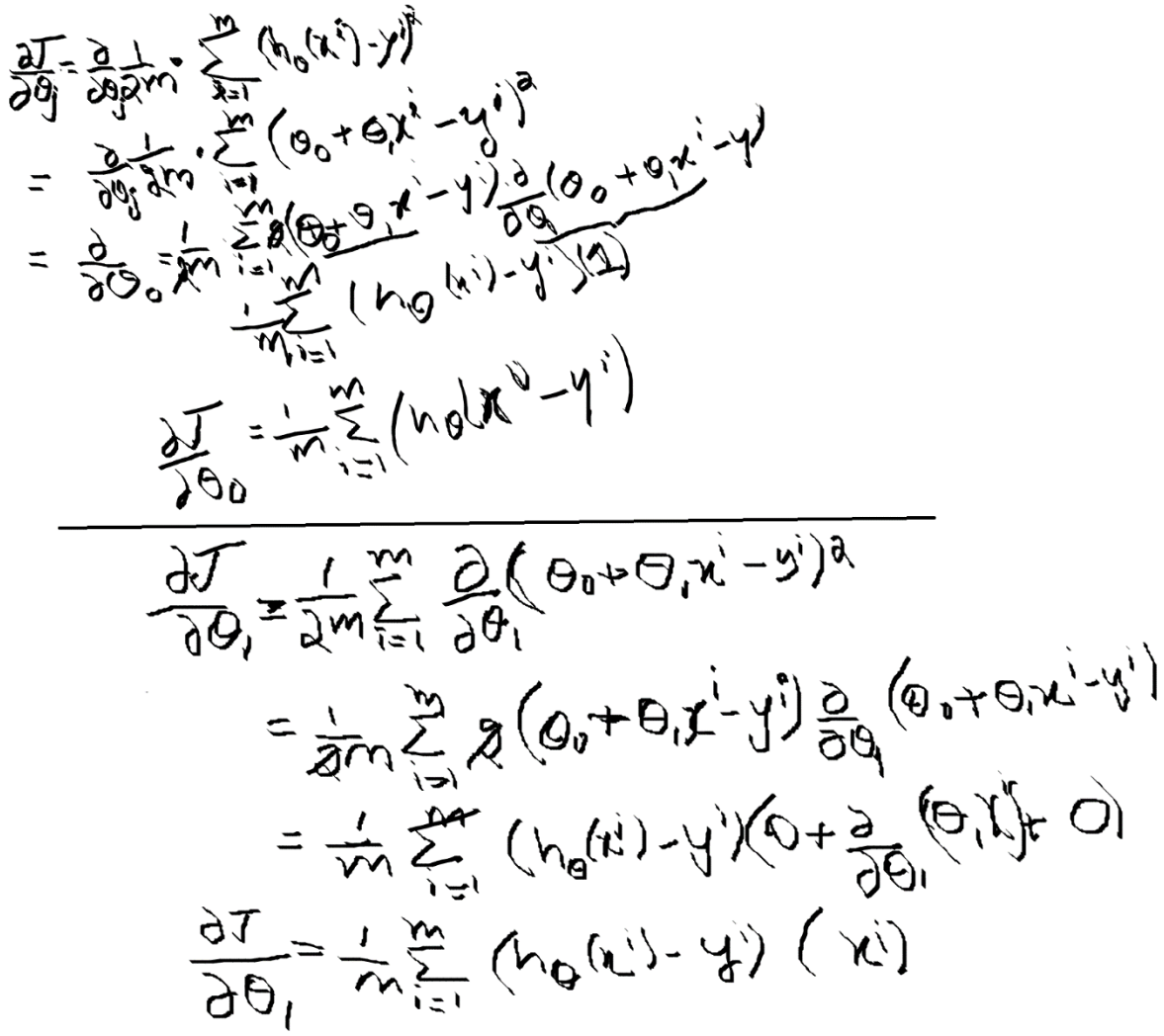
$$\theta_1 \;=\; \theta_1 - \alpha \frac{\partial}{\partial \theta 1}\, J(\theta_0, \theta_1)$$

}

The cost function for the linear Regression with Single variable is as follows

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

Now we calculate the two derivative functions as follows



This method looks at every example in the entire training set on every step and is called batch gradient descent.